Ein explizites Modell für die Fluid-Struktur-Interaktion basierend auf LBM und p-FEM

Von der
Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)
genehmigte

D i s s e r t a t i o n

von
Sebastian Geller
geboren am 02.11.1976
aus Wilhelm-Pieck-Stadt Guben

Eingereicht am	22. September 2009
Disputation am	01. März 2010
Berichterstatter	Prof. Dr.-Ing. habil. Manfred Krafczyk
	Prof. Dr.-Ing. Dieter Dinkler

2010

Wie puzzelt man ein Puzzle mit unendlich vielen Teilen,
bei dem keiner das Bild kennt?

Ein explizites Modell für die Fluid-Struktur-Interaktion basierend auf LBM und p-FEM

von Sebastian Geller

Zusammenfassung

Die Arbeit beschäftigt sich mit der partitionierten Kopplung von zwei effizienten Lösern zur Simulation von Fluid-Struktur-Interaktionsproblemen. Auf der einen Seite ist dies der auf der Lattice-Boltzmann-Methode basierende Fluidlöser VIRTUALFLUIDS und auf der anderen Seite der Finite-Elemente-Strukturlöser höherer Ordnung ADHOC. Anhand eines Benchmarks für poröse Medien wurde die Leistungsfähigkeit des Lattice-Boltzman-Lösers mit einem Navier-Stokes-Finite-Element-Löser verglichen, der auf den neuesten Techniken beruht. Die hohe Effizienz des Lösers motivierte die Untersuchung hinsichtlich einer Fluid-Struktur-Kopplung.

In dieser Arbeit ist es gelungen, ein explizites Kopplungsschema zu entwickeln. Dieses wird durch die schwach kompressible Formulierung der notwendigen Gleichungen ermöglicht. Notwendige Kraftinterpolationsverfahren wurden auf der Fluidseite entwickelt. Die sehr anspruchsvolle Behandlung von bewegten Geometrien auf quadtree/octreeartigen Gittern war eine Herausforderung des in der Arbeit entwickelten Gittergenerators, der für verschiedene Berechnungskerne generalisiert wurde. Die entwickelten Verfahren wurden anhand zahlreicher Benchmarks für zwei- und dreidimensionale Strömungsphänomene betrachtet. Untersuchungen von Starrkörperbewegungen halfen hierbei, das Kopplungsverfahren zu validieren. Ein numerischer und ein experimenteller Benchmark, initiiert von der DFG-Forschergruppe 493 (Fluid-Struktur-Wechselwirkung: Modellierung, Simulation, Optimierung), wurden ausführlich in 2-D validiert. In beiden Fällen befindet sich ein flexibles Fähnchen hinter einem angeströmten Zylinder und die resultierenden Auslenkungen sowie die Kräfte wurden mit anderen Forschungscodes verglichen. Für den numerischen Benchmark stimmten die Ergebnisse mit den Werten der anderen Codes überein, wohingegen beim experimentellen Benchmark die Schwingungsfrequenzen um bis zu 20 Prozent abwichen. Dies zeigte, dass die Simulationen mit dem expliziten Kopplungsschema durchgeführt werden können, aber sowohl verbesserte numerische Ansätze als auch neue Randbedingungen entwickelt und validiert werden müssen.

Für die dreidimensionale Fluid-Struktur-Interaktion führten die Testfälle einer in einem Rohr sinkenden Kugel und eine längsangeströmte Rechteckplatte zu guten Resultaten. Hierbei ist die Berechnungseffizienz der gekoppelten Simulation im Vergleich zu anderen impliziten Verfahren hervorzuheben. Kopplungsansätze mit der parallelen Version des dreidimensionalen Fluidströmungslösers wurden beim Testfall der sinkenden Kugel geprüft. Es wurde ein Löser entwickelt, der die Grundlage zur Berechnung von Fluid-Struktur-Phänomenen bildet. Für die realistische Anwendung müssen hier Verfahren der Turbulenzsimulation sowie der entwickelten Algorithmen zur Fluid-Struktur-Interaktion weiter validiert werden.

Der letzte Teil der Arbeit beschreibt die flexible Softwarearchitektur des VIRTUALFLUIDS-Strömungslösers, die auch für weitere physikalische Problemstellungen und eine interaktive Bedienung der laufenden Simulation entwickelt wurde. Es wird ein Computational-Steering-Ansatz beschrieben, der es dem Nutzer ermöglicht zur Berechnungslaufzeit interaktiv Simulationparameter zu ändern. Für den Anspruch Adaptivität, Parallelisierung und wissenschaftliche Visualisierung in einer Simulationsumgebung zu vereinen, ist der gezeigte Ansatz vielversprechend.

An explicit model for the fluid-structure interaction based on LBM and p-FEM

from Sebastian Geller

Abstract

This work deals with the partitioned coupling of two efficient solvers for the simulation of fluid-structure-interaction problems. The fluid solver VIRTUALFLUIDS, based on the Lattice-Boltzmann method, and the high order finite element structural solver ADHOC are used. Based on a benchmark for porous media, the efficiency of the Lattice-Boltzmann solver has been compared to a state of the art Navier-Stokes finite element solver. The resulting high efficiency has further motivated the development of the fluid-structure coupling approach.

In this work we succeessfully developed an explicit coupling scheme. This is possible due to the weakly compressible form of the governing equations. For the coupling, sophisticated force interpolation rules have been developed, in order to determine the fluid forces on the structures in the flow. Moreover, the handling of moveable geometries on quadtree/octree-typed grids was a major challenge for the grid generator. The developed methods have been validated with the help of numerous benchmark problems for two and three dimensional fluid-structure interaction phenomena. The study of rigid-body motion served to validate the coupling approach. For the two-dimensional case, a numerical and an experimental benchmark, initiated from the DFG Research Unit 493 (Fluid-structure interaction: modeling, simulation, optimization), were validated in detail. In both cases a flexible flag is attached to a cylinder in a flow field, and starts to move due to the asymmetric flow pattern. The resulting displacements as well as the forces are compared to results of other research groups. For the numerical benchmark the results agree well with the results from other codes, whereas for the experimental benchmark the frequency of the oscillations deviates up to 20 percentages. This points out that simulations with an explicit coupling scheme in general are feasible for the simulation of FSI, but nonetheless improved approaches as well as new boundary conditions have to be developed and validated.

For the validation of three dimensional fluid-structure interaction, a sinking sphere in a pipe and a rectangular plate in a cross flow are examined and lead to good results. For the sinking sphere case, the parallel version of the fluid solver VIRTUALFLUIDS has been used. Further three-dimensional applications indicate that for realistic simulations, the incorporation of turbulence models is crucial and the developed algorithms for fluid-structure interaction has to be validated. For all of the examined test cases, the high computational efficiency of the coupled simulation in comparison to implicit methods has to be pointed out.

The last part of this work describes the flexible software concept of the flow solver VIRTUALFLUIDS, which is the basis for the coupled computations of fluid-structure interaction phenomena. The solver was developed to be able to cope with further physical problem definitions and an interactive handling of the running simulation. In such a computational steering approach it is posssible for the user to interactivly change the simulation parameters during the computation. The presented software concept has shown to be capable of combining adaptivity, parallelization and scientific visualization in one simulation environment.

Danksagung

Die vorliegende Arbeit wurde während meiner Tätigkeit als wissenschaftlicher Mitarbeiter im Institut für rechnergestützte Modellierung der TU Braunschweig erstellt. An erster Stelle danke ich meinem Doktorvater Herrn Prof. Dr.-Ing. habil. Krafczyk für die engagierte Betreuung der Arbeit und die vielen fruchtbaren Diskussionen. Herrn Prof. Dr.-Ing. Dinkler danke ich für die Übernahme des Koreferats sowie Herrn Prof. Dr.-Ing. habil. Niemeier für den Vorsitz der Prüfungskommision.

Den Mitarbeiterinnen und Mitarbeitern des Lehrstuhls danke ich für ihre Unterstützung und Hilfsbereitschaft. Während meiner Tätigkeit teilte ich mit Sören Freudiger ein Büro und bedanke mich hiermit für die aktive Zusammenarbeit, das freundschaftliche Verhältnis und sein unterstützendes Fachwissen in den Tiefen der C++-Programmierung. Bei PD Dr.-Ing. habil. Jonas Tölke bedanke ich mich für die anregenden Diskussionen, die mir halfen meine Aufgabe zu meistern.

Des Weiteren möchte ich mich für die erfolgreichen und mitgestaltenden Studien- und Diplomarbeiten von Arne Mittelstaedt, Navodit Kaushik, Jan Linxweiler, Marco Schauer und Christian Janßen bedanken. Auch ist die Mithilfe meiner wissenschaftlichen Hilfskräfte Diana Buburuzan, Theodor Buburuzan, Jan Hegewald und Marco Schauer hervorzuheben, die einen wesentlichen Beitrag zur interaktiven Umgebung beisteuerten.

Da mein Projekt durch eine enge Kooperation mit dem Münchener Lehrstuhl für Computation in Engineering geprägt war, bedanke ich mich bei meinen Kollegen Dominik Scholz und Stefan Kollmannsberger. Insbesondere mit Stefan Kollmannsberger entwickelte sich neben der fachlichen Zusammenarbeit auch ein sehr freundschaftliches Verhältnis.

Für die konstruktiven Vorschläge beim Korrekturlesen meiner Arbeit bedanke ich mich bei Frank Molkenthin.

Zum Schluss danke ich meinen Eltern, die mir eine solide Ausbildung ermöglichten und immer ein wohliges Familienklima schufen. Ganz besonderer Dank gilt meiner Lebensgefährtin Christiane für ihre Geduld und Unterstützung sowie meiner Tochter Lena, die mit ihrem sonnigen Gemüt für die nötige Abwechslung sorgte.

Diese Arbeit wurde von der Deutsche Forschungsgemeinschaft (DFG) gefördert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Tacoma, US -Bundesstaat Washington, 07.11.1940: Wie so oft wiegt sich die erst kürzlich eingeweihte Hängebrücke über dem Puget Sound im Wind, der an Stärke langsam zunimmt und eine Geschwindigkeit von 68 km/h erreicht. Plötzlich werden die Schwingungen immer größer, einige Stahlseile reißen, das Deck bricht auseinander und gewaltige Teile stürzen in die Tiefe. [92] - Ein Beispiel von vielen, die aufgrund von Naturgewalten traurige Berühmtheit erlangten.

Ziel der Forschung muss sein, Baukonstruktionen zu ermöglichen, die von Menschen nicht steuerbare Einflüsse berücksichtigen. Die rechnergestützte Simulation von Phänomenen, die Fluid-Struktur-Interaktion (FSI) beinhalten, ist eine Methode, die den Einfluss von Wasser und Wind auf Bauwerke untersucht und somit ihren Beitrag dazu leistet. Beispiele für das Bauingenieurwesen sind u. a. Brücken, Wolkenkratzer und Offshoreplattformen, die bekanntlich hohen Windlasten ausgesetzt sind. Im Maschinenbau kann man Ventile, Pumpen, Flügel, Schiffspropeller und ähnliches mit FSI-Berechnungen dimensionieren. Angewendet wird FSI ebenfalls beim Design von Rührern, Extrudern, Einspritzsystemen in der Verfahrenstechnik sowie bei künstlichen Herzklappen und Blutgefäßen in der Medizintechnik.

Als Fluid-Struktur-Interaktion (FSI) wird die Kopplung einer numerischen Strömungsberechnung (CFD) mit einer Berechnung der Strukturverformung (CSD) bezeichnet. Dies tritt in Fällen auf, wo Strömungen das Bauteil durch Druck- und Reibungskräfte bzw. thermische Einflüsse verformen. Bekannte Fluid-Struktur-Phänomene sind das Flattern (Biege- und Torsionsschwingung) und Galloping. Flattern tritt vor allem bei Düsen oder Turbinen in der Aerodynamik auf und Galloping findet man bei Kabeln, Brücken oder Offshorewindkraftanlagen.

Simulationen sind bedeutsam, um Vorhersagen zu treffen, Abläufe zu verstehen und Bauteile auszulegen. Experimentelle Untersuchungen sind häufig wegen ihres hohen Risikoanteils und den beträchtlichen Kosten nicht ausführbar. Ein weiteres Problem bei Experimenten ist die immense Schar an Modellparametern. Hierbei kann die Computersimulation Abhilfe schaffen. Diese gibt Aufschluss über die Validierung von Designkonzepten und/oder Fertigungsprozesse. Damit sollen Modifikationen oder ein Redesign im fortgeschrittenen Entwicklungsprozess vereinfacht und die damit verbundenen Kosten reduziert werden.

Die realistische, transiente Simulation von Fluid-Struktur-Interaktionsprozessen ist eine Herausforderung im Hinblick auf die Hardwarekapazität und die numerische Technik. Zusätzlich zum Aufwand, resultierend aus der Kombination der Struktur- und Fluidberechnung, führt die Interaktion zu einer Vielfalt von komplexen numerischen und mechanischen Fragen. Dies umfasst Fragen der entsprechenden Betrachtung der dynamischen und nichtlinearen Interaktion selbst, die Möglichkeit von partitionierten Lösungsansätzen, die Tauglichkeit von bewährten Methoden von der Struktur- und Strömungsseite, die Robustheit und Effizienz der Methoden und die entsprechende Beschreibung von komplexen, zeitlich veränderlichen Geometrien.

Fluid-Struktur-Interaktion ist ein faszinierendes Thema der Forschung. Es ist schwer, eine gute Lösung zu erhalten, da eine Simulation eine große Anzahl von Modellannahmen mit einschließt. Diese werden hierbei in der Gesamtkette von der Problemstellung bis zur Auswertung getroffen: Ein Problem mit seinen Bestandteilen und Wechselwirkungen führt mit Hilfe der Ordnungsprinzipien wie Erhaltungssätze, Kräftegleichgewichte, Symmetrie, etc. und der Approximation mit den entsprechenden Differentialgleichungen sowie den zugehörigen Rand- und Anfangsbedingungen, der entsprechenden Numerik und den Algorithmen sowie der Implementierungsweise zu Simulationsergebnissen, die ausgewertet werden müssen. Bei der Lösung wird diese Modellkette als iterativer Prozess durchlaufen. Das schwächste Glied der Kette ist hierbei maßgebend für die Qualität des Ergebnisses. Bei der Fluid-Struktur-Interaktion ist über die Modellfehler und deren gegenseitige Einflüsse wenig bekannt. Daher ist die Validierung von Fluid-Struktur-Simulationen ein bedeutender Aspekt für die Forschung.

Um Einblick in die zugrundeliegende Physik zu erhalten und Optimierungen durchzuführen, ist die numerische Berechnung der gekoppelten Fluid-Struktur-Simulation unabdingbar. Es ist aber heutzutage immer noch nicht möglich, spezifische Strömungsphänomene am Computer allgemein abzubilden. Dafür sind teure Windkanal- bzw. Laborexperimente nötig. Für die Untersuchung des strömungstechnischen Problems werden größtenteils die Navier-Stokes-Gleichungen numerisch mit den verschiedensten Verfahren diskretisiert (FVM, FEM, FDM). Als alternativer Ansatz zur Strömungssimulation etablierte sich in den letzten Jahren das Lattice-Boltzmann-Verfahren. Dieses löst im Gegensatz zum erstgenannten Verfahren das Strömungsproblem aus der mesoskopischen Sichtweise und beruht auf dem stochastischen Verhalten hypothetischer Gasteilchen.

Im Allgemeinen existieren zwei Klassen von Ansätzen zur Lösung von Fluid-Struktur-Interaktionsproblemen. Der monolithische Ansatz [50, 52, 136] diskretisiert die beiden separaten Bereiche mit dem gleichen Diskretisierungsschema und löst das resultierende Gesamtgleichungssystem. Die Kompatibilitätsbedingung am Interface sind im Gleichungssystem enthalten. Im Gegenzug nutzt der partitionierte Ansatz [124] separate Löser für das Fluid- und Strukturgleichungssystem. Es existieren starke Kopplungsmethoden [30, 126, 137] sowie schwache Kopplungsmethoden [13, 77]. In beiden Fällen müssen die Löser an ihrem gemeinsamen Rand kommunizieren, um die Interfacebedingungen iterativ zu erfüllen. In einem partitionierten Ansatz kann sowohl das Strömungs- als auch das Strukturgebiet für die entsprechenden Felder individuell diskretisiert werden. Für das Fluid existieren hauptsächlich zwei Ansätze im Kontext von partitionierter FSI mit großen Deformationen. Das Fluid ist entweder diskretisiert als (a) beliebig bewegliches Netz mit der sogenannten Arbitrary Lagrangian Eulerian (ALE) Formulierung oder (b) mit einem ortsfesten kartesischen Gitter (eingebetteter Ansatz/Euler-Ansatz) [83]. Der Vorteil der ALE Methode ist die konsistente Behandlung des Interfaces und die Vermeidung der Aktivierung/Deaktivierung von Fluidknoten infolge des bewegten Strukturnetzes. Der größte Nachteil dieses Ansatzes ist die Neuvernetzungsprozedur beim Auftreten großer Verformungen [124]. Ein für dieses Problem verwendeter Ansatz ist ein Schema, bei dem ein festes Euler'sches Fluidnetz, das nicht zur Strukturdomaine gehört, überlappt wird durch ein bewegliches deformierbares Fluidgitter, das in der ALE-Formulierung beschrieben ist. Solche Schemen wurden z. B. in [39, 138] vorgeschlagen. Zur Vollständigkeit sei noch die Partikel Element Methode [53] genannt werden, mit der FSI-Probleme mit großen Verformungen berechnet werden können. In beiden Fällen muss die Geometrie des Interfaces explizit beschrieben werden. Vom geometrischen Gesichtspunkt ist der Hauptunterschied zwischen den Methoden (a) und (b), dass für die ALE-Formulierung die Bewegung des Randes sich im Fluidgebiet ausbreitet, d.h. die Fluiddiskretisierung muss der Randbewegung nicht nur am Rand, sondern auch im inneren Fluidgebiet folgen. Ungeachtet von der zugrundeliegenden Diskretisierungsmethode (Finite Elemente, Finite Volumen) ist das Fluid gewöhnlich durch ein Netz diskretisiert, welches bestimmte geometrische Bedingungen (z. B. Elementgröße und Aspect Ratio) erfüllen muß. Dies schränkt im Gegenzug die Bewegung des Randes ein. Schwierigkeiten entstehen beispielsweise in Situationen, wo Randverformungen sehr groß sind, d.h. die Netzdeformationen des betrachteten Fluidgebietes sind so stark, dass eine Neuvernetzung erforderlich ist. Speziell in drei Raumdimensionen stellt dies eine große Herausforderung dar.

Die Behandlung von großen Verschiebungen [77] ist der große Vorteil von den Methoden, die auf ortsfesten kartesischen Gittern operieren. Die Schwierigkeit hierbei liegt darin, die Interfacebedingungen zwischen Struktur und Fluid richtig abzubilden. Eine gute Übersicht sowie ein Vergleich zwischen den Methoden (a) und (b) ist in [83] gegeben.

Der Lattice-Boltzmann-Ansatz [100] wird als eingebetteter Ansatz kategorisiert, da er auf ortsfesten kartesischen Gittern basiert. Eine lokale Vefeinerung kann beispieelsweise durch quadtreeartige Gitter in 2-D oder octreeartige Gitter in 3-D erreicht werden [17, 29, 140]. Der Extraaufwand zur Behandlung bewegter Ränder über einem orstfesten Gitter liegt in der Berechnung der Gitterabstände zum bewegten

Rand und ist begrenzt auf Regionen nahe zum Rand (siehe Kapitel 2.4). Die Anpassung des Quad-/Octrees benötigt jedoch nur lokale Operationen wodurch eine zeitraubende Neuvernetzungsprozedur vermieden wird.

Für einen partitionierten Ansatz ist es unabdingbar, dass beide Löser (CFD und CSD) effizient sind. In dieser Arbeit wird eine Kopplung zwischen einem LB-Löser, basierend auf adaptiven quad- und octreeartigen Knoten- und Blockgittern und einem p-FEM Strukturlöser untersucht. Die Kopplung wird durch ein bewegtes Interface-Netz, das in Kapitel 4.3 beschrieben ist, erhalten.

Beide Löser haben sich als sehr effizient in ihrem Feld erwiesen. Der Finite Element Löser hoher Ordnung hat exponentielle Konvergenzraten in der Energienorm für glatte Probleme. Schnelle und genaue Lösungen im zwei oder dreidimensionalen Raum sind ebenfalls für geometrisch nichtlineare Probleme vorhanden. Die Effizienz und Genauigkeit des CSD-Lösers ADHOC [22] im Kontext der Fluid-Struktur-Interaktion wird in [108, 109] diskutiert. Die Effizienz des CFD-Lösers VIRTUALFLUIDS wurde in [37] demonstriert und wird in den nachfolgenden Kapiteln beschrieben. Der Lattice-Boltzmann-Strömungslöser basiert auf einem expliziten und schwach kompressiblen Schema, aufgrund dessen ein allgemeines explizites Kopplungsschema entwickelt werden kann. Das Fluid wird auf einem adaptiven, kartesischen Gitter unabhängig von hohen Gitterdeformationen und den hiermit assoziierten Problemen diskretisiert.

Die ersten Lattice-Boltzmann-Algorithmen für Fluid-Struktur-Interaktionsprobleme wurden von Ladd [70, 71] für die Simulation von Partikelsuspension entwickelt. Eine spezielle Behandlung der Randbedingungen und die Aktivierung/Deaktivierung von Fluidknoten werden in dieser Arbeit gezeigt. Krafczyk et al. [67] erweiterten diese Ideen zur gekoppelten zweidimensionalen Simulation von künstlichen Herzklappen. Während hier gute Ergebnisse für die Verschiebungen erzielt wurden, zeigte der zeitliche Verlauf der Anström- und Auftriebskräfte den Einfluss der Aktivierung und Deaktivierung neuer Fluidknoten. Die Strömung in dem Kanal wurde durch eine transiente Randbedingung, die ein physiologisch relevantes Regime repräsentiert, angetrieben. Shi und Phan-Thien [115] führten die Distributed-Lagrange-Multiplier/Fictitious-Domain (DLM/FD) Methode ein, um elastische Verformungen mittels Fluid-Struktur-Interaktion mit der Lattice-Boltzmann-Methode zu behandeln. In [113] wurde dieser Ansatz auf drei Raumdimensionen erweitert. Eine Immersed-Boundary-Lattice-Boltzmann-Methode zur Lösung von Fluid-Partikel-Interaktionen wurde in [28, 27, 44] entwickelt. Die Methode wurde in [120] durch eine Multi-Block-Strategie, die eine Gitterverfeinerung erlaubt, weiterentwickelt. In dieser Arbeit werden die Kopplung mit dem Standard-Lattice-Boltzmann-Verfahren und die hierfür notwendigen Anpassungen der Methode vorgestellt.

Im Rahmen dieser Arbeit entstand die Basis eines adaptiven, baumartigen Fluidlösers, der verschiedene Rechenkerne mit derselben Gebietsdiskretisierung umfasst. Das Softwareframework basiert auf objektorientierten Technologien und trägt den Namen VIRTUALFLUIDS. Der entwickelte Entwurf für den Softwareprototypen war ein entscheidener Faktor, der zum Erfolg der komplexen Simulationen führte. Für den Simulationsprozess ist es außerdem wichtig, dass die Simulation interaktiv abläuft und Simulationsparameter zur Laufzeit geändert werden können. Es wird gezeigt, dass die klassische Schleife von Präprozessing, Berechnung und Postprozessing durch eine interaktive Umgebung durchbrochen werden kann. Normalerweise bedeutet ein getrenntes Prä- und Postprozessing, dass die Eingabedaten jeweils manuell eingebunden werden. Im Designzyklus kann dies wertvolle Bearbeitungszeit in Anspruch nehmen. Es wird ein Computational-Steering-Ansatz eingeführt, bei dem der Nutzer während des Berechnungsprozesses interaktiv Parameter modifizieren, Randbedingungen ändern, Geometrien entfernen, hinzufügen oder verschieben kann.

Der Ausdruck Computational Steering wurde von Liere et al. [78] eingeführt und beschreibt die interaktive Modifizierung von Parametern während des Simulationsprozesses. Dem Nutzer wird ermöglicht,

die Eingabeparameter des berechnenden Prozesses zu ändern und dessen Reaktion zur Laufzeit zu studieren. Der Ingenieur, der eine steuerbare Simulation durchführt, erhält ein intuitiveres Verständnis von dessen Verhalten durch das direkte Wechselspiel zwischen Modifikation und Reaktion des Prozesses. Neben den grundlegenden Anforderungen der einfachen Anpassungsfähigkeit und Erweiterbarkeit für neue strömungsmechanische Probleme, zielt die Softwarebibliothek auch auf physikalische Korrektheit, Parallelisierung und eine einfache, intuitive Anwendung ab.

Übersicht

Die Arbeit gliedert sich in drei Teile. Im ersten Teil (Kapitel 2-4) wird das Fluid- und Strukturmodell, das zugrunde liegende Berechnungsgitter, die notwendigen Erweiterungen für die FSI und der Kopplungsansatz beschrieben. In Kapitel 2.1 werden zunächst die Grundlagen der Lattice-Boltzmann-Methode diskutiert, worauf die Beschreibung des Multiple-Relaxation-Time(MRT)-Modells in Kapitel 2.1.2 folgt. Im nächsten Abschnitt wird eine detailierte Diskussion über Randbedingungen und deren Implementierung für bewegte Ränder geführt. In Kapitel 2.2.1 wird die Vorgehensweise bei der Aktivierung/Deaktivierung von Fluidknoten gezeigt. Danach folgen die Kapitel über die Gitterverfeinerung und -generierung.

Da der Fokus der Arbeit auf der Entwicklung des Fluidlösers liegt, werden in Kapitel 3 die zugrundeliegenden Gleichungen für das Strukturproblem und der entsprechende Computational Structural Dynamics (CSD) Löser kurz skizziert.

Zu Beginn des vierten Kapitels (Abschnitt 4.1) wird die Berechnung der Kräfte auf feste und bewegliche Strukturen beschrieben und in 4.2 die Transformation der physikalischen Größen zwischen dem Lattice-Boltzmann-System und dem realen System. Der explizite und implizite Kopplungsalgorithmus für die Fluid-Struktur-Interaktion werden in Kapitel 4.3 präsentiert und schließen den Grundlagenteil ab.

Im zweiten Teil, dem Validierungsteil, werden zunächst Testfälle im zweidimensionalen Raum untersucht. Der Benchmark für poröse Medien in Kapitel 5.1 zeigt die Berechnungseffizienz der Lattice-Boltzmann-Methode für transiente Problemstellungen. Danach wird eine Starrkörperbewegung eines Einmassenschwingers in Kapitel 5.2 mit Literaturergebnissen verglichen. Der am intensivsten validierte Testfall ist der numerische Benchmark eines schwingenden Fähnchens hinter einem Zylinder in Kapitel 5.3, der in der DFG-Forschergruppe 493 definiert wurde. Der letzte Testfall in 2-D ist der experimentelle Benchmark in Kapitel 5.4, der ebenfalls im Rahmen der Forschergruppe untersucht wurde. Im Kapitel 6.1 werden in 3-D die Druckbeiwerte eines Zylinders berechnet. Als erster Testfall für eine Starrkörperkopplung dient eine sinkende Kugel in Kapitel 6.2. Die Kopplung mit dem Strukturlöser ADHOC wird in Kapitel 6.3 anhand einer längsangeströmten Rechteckplatte, bei der für verschiedene Anströmgeschwindigkeiten die Auslenkungen verglichen werden, untersucht.

Im dritten Teil (Kapitel 7) wird die, in dieser Arbeit entwickelte, flexible Softwarearchitektur vorgestellt. Neben der Erläuterung der grundlegenden Struktur wird auch auf die Parallelisierung und die Aspektorientierung eingegangen.

Den Abschluss bilden die Schlussfolgerungen und der Ausblick.

2 Lattice-Boltzmann-Methode zur Fluidsimulation

In dieser Arbeit kommt zur numerischen Simulation der Fluiddynamik das Lattice-Boltzmann-Verfahren zum Einsatz, das eine sehr effiziente Beschreibung des Fluids in mesoskopischer Betrachtungsweise ermöglicht. Im nachfolgenden werden die dafür benötigten Grundgleichungen, Randbedingungen und Gebietsdiskretisierungen eingeführt.

2.1 Grundlagen der Lattice-Boltzmann-Methode

2.1.1 Einleitung

Während der letzten zwei Jahrzehnte war ein großer Fortschritt bei der Entwicklung kinetischer Modelle, speziell der Lattice-Boltzmann-Methode (LBM), für Strömungsprobleme festzustellen. Für eine Einführung in die Thematik wird auf [45, 139, 119] und die Literaturhinweise darin verwiesen.

Die Grundgleichung der kinetischen Gastheorie ist die Boltzmanngleichung, welche die Dynamik einer Partikelverteilungsfunktion f (Einheit: $\frac{kg}{m^3}$) in Zeit t und Phasenraum $\mathbf{x}, \boldsymbol{\xi}$ beschreibt, wobei $\mathbf{x}$ die Position und $\boldsymbol{\xi}$ die mikroskopische Geschwindigkeit ist:

$$\partial_t f(t, \mathbf{x}, \boldsymbol{\xi}) + (\boldsymbol{\xi} \cdot \boldsymbol{\nabla}_x) f(t, \mathbf{x}, \boldsymbol{\xi}) = \Omega(f) \tag{2.1}$$

Die linke Seite von Gleichung 2.1 beschreibt die Advektion der Verteilungen mit der Geschwindigkeit $\boldsymbol{\xi}$ und der Kollisionsterm Ω auf der rechten Seite die Interaktion der Partikelverteilungen. Der Kollisionsterm Ω wird durch ein komplexes Integral beschrieben, dessen genaue Lösung nicht bekannt ist. Daher wird der Kollisionsterm mit vereinfachten Modellen, hier dem Bhatnagar, Gross und Krook Modell (BGK) [5] angenähert.

$$\Omega = -\frac{1}{\tau}(f(t, \mathbf{x}, \boldsymbol{\xi}) - f^{eq}(t, \mathbf{x}, \boldsymbol{\xi})) \tag{2.2}$$

$f^{eq}(t, \mathbf{x}, \boldsymbol{\xi})$ ist die Taylorexpansion der Maxwellverteilung, die die Gleichgewichtsverteilung für ein ideales Gas ist und τ (Einheit: s) die Relaxationszeit. Die Differenz $f - f^{eq}$ entspricht dem Nichtgleichgewichtsanteil f^{neq}. Unter der Annahme kontinuierlicher Strömungen mit kleinen Knudsenzahlen ($Kn << 1$) kann der Geschwindigkeitsraum durch einen Satz $b + 1$ diskreter Geschwindigkeiten $\{\mathbf{e}_i, i = 0, \ldots, b\}$ angenähert werden. In dieser Arbeit wird das sogenannte D2Q9-Modell in 2-D und das D3Q19-Modell in 3-D verwendet. Die Notation DdQq folgt hierbei [100], in der d die Anzahl der Dimensionen und q die Anzahl der diskreten Geschwindigkeiten ist.

Die Richtungsvektoren sind für das D2Q9-Modell:

$$\{\mathbf{h}_i\} = \left\{ \begin{array}{ccccccccc} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{array} \right\} \tag{2.3}$$

und für das D3Q19-Modell:

$$\{\mathbf{h}_i\} = \left\{ \begin{array}{cccccccccccccccccc} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{array} \right\} \tag{2.4}$$

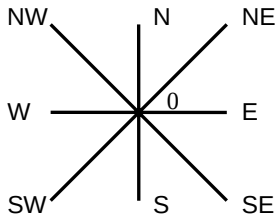


Abbildung 2.1: *D2Q9-Modell*

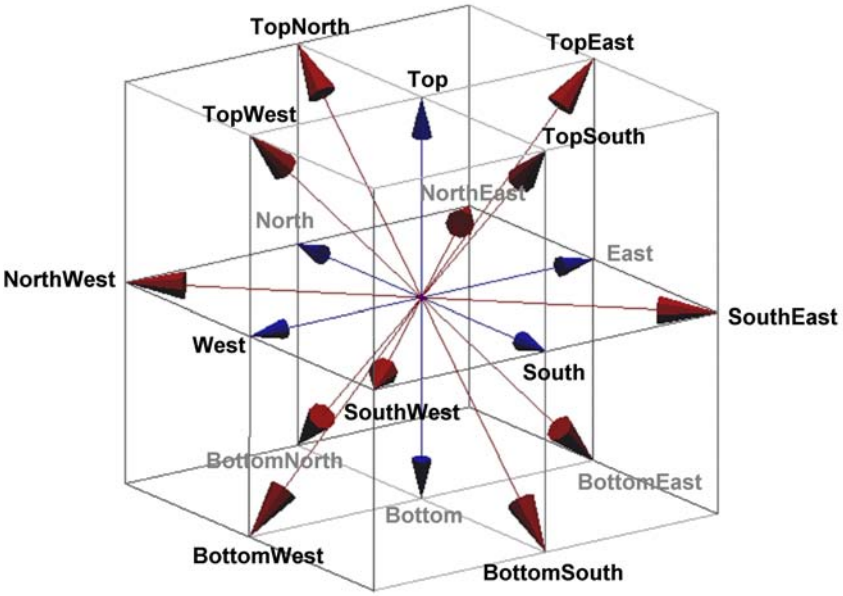


Abbildung 2.2: *D3Q19-Modell*

Die diskreten Geschwindigkeiten sind $\mathbf{e}_i = \boldsymbol{\xi}_i = c \cdot \mathbf{h}_i$. Die konstante Geschwindigkeit c (Einheit: $\frac{m}{s}$) korreliert mit der Schallgeschwindigkeit gemäß $c_s^2 = c^2/3$. Diese verknüpft mit der Dichte ergibt den Druck $p = c_s^2 \rho$.

Nach der Projektion des Kollisionsoperators in den Geschwindigkeitsraum erhalten wir das diskrete Geschwindigkeitsmodell bzw. die diskrete Boltzmanngleichung [48] bestehend aus $b + 1$ Gleichungen:

$$\partial_t \, f_i(t, \mathbf{x}) + (\mathbf{e}_i \cdot \boldsymbol{\nabla}_x) f_i(t, \mathbf{x}) = \Omega_i, \quad i = 0, \ldots, b \tag{2.5}$$

mit den Verteilungsfunktionen $\boldsymbol{f} = (f_i(\mathbf{x}, t), i = 0 \ldots, b)$ (Einheit: $\frac{kg}{m^3}$), welche mit der Geschwindigkeit $\mathbf{e}_i$ propagieren.

Die Gleichgewichtsverteilungsfunktionen f_i^{eq}, auch Gleichgewichtsverteilungen genannt, werden gewöhnlich als quadratische Polynome der ersten beiden hydrodynamischen Momente [100] – der Dichtefluktuation ρ und des Impulses $\rho_0 \mathbf{u}$ gewählt. Diese sind folgendermaßen mit den Verteilungen verknüpft:

$$\rho = \sum_{i=0}^{b} f_i \qquad \rho_0 \mathbf{u} = \sum_{i=0}^{b} \mathbf{e}_i f_i \tag{2.6}$$

Die Gleichgewichtsverteilungen, hergeleitet für inkompressible Strömungen [47, 59], sind:

$$f_i^{eq} = w_i \left(\rho + \rho_0 \left(3 \frac{\mathbf{e}_i \cdot \mathbf{u}}{c^2} + \frac{9}{2} \frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2} \frac{u^2}{c^2} \right) \right), \tag{2.7}$$

mit der konstanten Referenzdichte ρ_0 (normalerweise $= 1$) und den Wichtungsfaktoren w_i für das jeweilige Modell, wobei die Richtungen i den englischen Bezeichnungen **E**ast, **N**orth, **S**outh, **W**est, **T**op, **B**ottom für die diskreten Gittervektoren entsprechen.

Für das D2Q9-Modell sind die Wichtungsfaktoren [45, 66]:

$$w_i = \begin{cases} 4/9, & i = 0 \\ 1/9, & i = E, W, N, S \\ 1/36, & i = NE, SW, SE, NW \end{cases} \tag{2.8}$$

und für das D3Q19-Modell:

$$w_i = \begin{cases} 1/3, & i = 0 \\ 1/18 & i = E, W, N, S, T, B \\ 1/36, & i = NE, SW, SE, NW, TE, BW, BE, TW, TN, BS, BN, TS \end{cases} \tag{2.9}$$

Diskretisiert man Gleichung 2.5 mittels eines Upwind-Finite-Differenzen-Ansatzes, gekoppelt in Raum und Zeit durch $\Delta x = c\Delta t$, erhält man die Evolutionsgleichung auch genannt Lattice-Boltzmann-Gleichung [48]

$$f_i(t + \Delta t, \mathbf{x} + \mathbf{e}_i \Delta t) - f_i(t, \mathbf{x}) = \Omega_i, \quad i = 0, \ldots, b \tag{2.10}$$

auf einem raumfüllenden Gitter mit dem Knotenabstand $\Delta x = c\Delta t$.

Der Kollisionsoperator ist nun

$$\Omega_i = -\omega \left(f_i - f_i^{eq} \right) \tag{2.11}$$

mit $\omega = \frac{\Delta t}{\tau}$.

Im Gegensatz zum einfachen FD-Ansatz sind auch andere Diskretisierungstechniken möglich, wie z. B. in [75, 91, 114, 132, 131, 130] gezeigt wurde. Für die FD-Diskretisierung ist die Kollisionsrate ω folgendermaßen mit der kinematischen Viskosität ν (Einheit: $\frac{m^2}{s}$) verknüpft:

$$\nu = c^2 \Delta t \left(\frac{1}{3\omega} - \frac{1}{6} \right) \tag{2.12}$$

Mittels einer Chapman-Enskog-Analyse [15, 34] kann gezeigt werden, dass die Momente (2.6) der Lösung von Gleichung 2.10 Lösungen der inkompressiblen Navier-Stokes-Gleichungen bis zu einem Fehler der Ordnung $\mathcal{O}\left(\Delta x^2\right), \mathcal{O}\left(Ma^2\right)$ sind.

Prinzipiell lässt sich die Lattice-Boltzmann-Gleichung in zwei Schritte aufteilen, den Kollisionsschritt und den Propagationsschritt. Für uniforme Gitter ergibt sich somit ein recht einfacher Algorithmus, um die Strömung zu simulieren. Es werden zuerst für alle Knoten die neuen Verteilungen nach der Kollision bestimmt und im nächsten Durchlauf zu ihren Nachbarknoten verschoben.

Ein typisches Schema der Implementierung ist, die Gleichung 2.10 in einen lokalen Kollisionschritt

$$\tilde{f}_i(t + \Delta t, \mathbf{x}) = f_i(t, \mathbf{x}) + \Omega_i(t, \mathbf{x}), \tag{2.13}$$

mit den sogenannten Post-Kollisionwerten $\tilde{f}_i(t + \Delta t, \mathbf{x})$ und einen Propagationsschritt, bei dem die Verteilungen zu ihren Nachbarknoten bewegt (propagiert) werden, aufzusplitten.

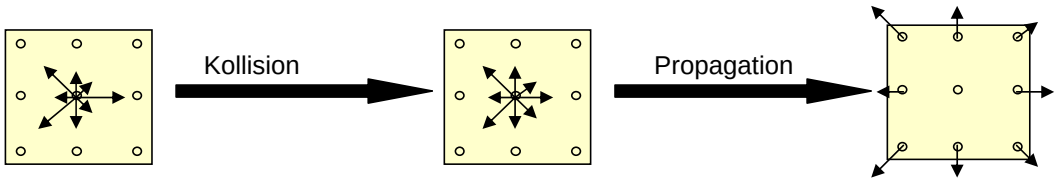


Abbildung 2.3: *Kollisions- und Propagationsschritt*

2.1.2 Momentenmethode

Im sogenannten Multiple-Relexation-Time(MRT)-Modell werden die Verteilungen mittels einer Transformationsmatrix M, bestehend aus orthogonalen Eigenvektoren des Kollisionsoperators, in den Momentenraum transformiert [19, 20, 73]. Vereinfachend werden hier nur die Momente für das D2Q9-Modell gezeigt. Für das D3Q19-Modell können sie Veröffentlichung [129] entnommen werden.

$$\boldsymbol{m} = \mathsf{M}\boldsymbol{f} \quad \boldsymbol{f} = \mathsf{M}^{-1}\boldsymbol{m} \tag{2.14}$$

Die Momente $\boldsymbol{m}$ werden folgendermaßen definiert:

$$\boldsymbol{m} = (\rho,\ e,\ \epsilon,\ \rho_0 u_x,\ q_x,\ \rho_0 u_y,\ q_y,\ p_{xx},\ p_{xy}) \tag{2.15}$$

Die Momente 2. Ordnung e, p_{xx}, p_{xy} korellieren mit dem Spannnungstensor $\sigma_{\alpha\beta}$:

$$\sigma_{11} = -\left(1-\frac{\omega}{2}\right)\left(\frac{1}{3}\rho\, c^2 + \frac{1}{2}p_{xx} + \frac{1}{6}e - \rho_0 u_x^2\right) \tag{2.16}$$

$$\sigma_{22} = -\left(1-\frac{\omega}{2}\right)\left(\frac{1}{3}\rho\, c^2 - \frac{1}{2}p_{xx} + \frac{1}{6}e - \rho_0 u_y^2\right) \tag{2.17}$$

$$\sigma_{12} = -\left(1-\frac{\omega}{2}\right)\left(p_{xy} - \rho_0 u_x u_y\right), \tag{2.18}$$

wobei ω der Relaxationsrate entspricht.

Die restlichen Größen sind die Dissipation ϵ und die Wärmeflüsse q_x, q_y.

Die Transformationsmatrix lautet:

$$\mathsf{M} = \begin{bmatrix} 1\cdot & (1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\ c^2\cdot & (-4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2) \\ c^4\cdot & (4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1) \\ c\cdot & (0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1) \\ c^3\cdot & (0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1) \\ c\cdot & (0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1) \\ c^3\cdot & (0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1) \\ c^2\cdot & (0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) \\ c^2\cdot & (0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) \end{bmatrix} \tag{2.19}$$

Beim MRT-Modell wird die Kollision im Momentenraum ausgeführt. Sie ist gegeben durch

$$\boldsymbol{\Omega} = -\mathsf{M}^{-1}\mathsf{S}\left[(\mathsf{M}\boldsymbol{f}) - \boldsymbol{m}^{eq}\right], \tag{2.20}$$

mit den Gleichgewichtsmomenten $\boldsymbol{m}^{eq}$:

$$\begin{aligned}
m_0^{eq} &= \rho \\
m_1^{eq} &= e^{eq} = -2c^2\rho + 3\rho_0 u^2 \\
m_2^{eq} &= \epsilon^{eq} = c^4\rho - 3c^2\rho_0 u^2 \\
m_3^{eq} &= \rho_0 u_x \\
m_4^{eq} &= q_x^{eq} = -c^2\rho_0 u_x \\
m_5^{eq} &= \rho_0 u_y \\
m_6^{eq} &= q_y^{eq} = -c^2\rho_0 u_y \\
m_7^{eq} &= p_{xx}^{eq} = \rho_0(u_x^2 - u_y^2) \\
m_8^{eq} &= p_{xy}^{eq} = \rho_0 u_x u_y.
\end{aligned} \tag{2.21}$$

S ist eine Diagonalmatrix mit den einzelnen Relaxationssraten s_i.

Die Kollisionsraten s_0, s_3, s_5 sind nicht relevant, da sie mit den konservativen Momenten verknüpft sind. Um eine konsistente dynamische Viskosität (2.12) zu erhalten, entsprechen Raten s_7 und s_8 der BGK-Kollisionsrate $s_7 = s_8 = \omega$.

Die anderen Relaxationsraten haben keine physikalische Bedeutung für inkompressible Strömungen und können im Bereich $[0, 2]$ frei gewählt werden. Durch eine geeignete Wahl kann man die Stabilität des Verfahrens verbessern [73]. Die Wahl von $\mathsf{S} = diag(\frac{1}{\tau})$ reduziert das MRT-Modell zum BGK-Modell, da die Relaxationsraten für die nichterhaltenden Momente identisch sind. Die optimalen Werte hängen vom spezifischen System, d.h. von Geometrie, Initial- und Randbedingungen ab und können i. A. nicht a priori bestimmt werden. In [20, 73] werden sinnvolle Werte vorgeschlagen. Beim Two-Relaxation-Time (TRT) Modell, dass von Ginzburg [41] entwickelt wurde, existieren nur zwei Kollisionsraten. Die ungeraden Momente werden dabei mit $s_4 = s_6 = 2 - \omega$ relaxiert, die anderen mit ω. Hier wurde $s_1 = s_2 = s_4 = s_6 = 1$ gewählt. Dies bedeutet, dass die dazugehörigen Momente sofort zu ihren Gleichgewichtswerten relaxieren. Dies wurde auch von Yu [141] vorgeschlagen.

Der Basisalgorithmus der MRT-Kollision für jeden Gitterknoten und Zeitschritt lautet:

Algorithmus 1 MRT-Kollision

- $\Rightarrow$ transformiere die Knotenverteilungen in einen äquivalenten Satz von Momenten $(\boldsymbol{m} = \mathsf{M}\boldsymbol{f})$
- $\Rightarrow$ relaxiere $b - 3$ (in 2-D) nicht-erhaltene Momente m_k mit verschiedenen Relaxationsraten $\tau_k = \frac{1}{\omega_k}$ in Richtung der Gleichgewichtsmomente m_k^{eq} (siehe [73])
- $\Rightarrow$ transformiere die relaxierten Momente $\left(\tilde{\boldsymbol{f}} = \mathsf{M}^{-1}\tilde{\boldsymbol{m}}\right)$ zurück in den Verteilungsraum um die Verteilungen im Post-Relaxationszustand zu erhalten

Mittels einer Chapman-Enskog-Analyse [15, 34] oder einer asymptotischen Analyse [58, 60] kann gezeigt werden, dass die Konvergenzordnung im Hinblick zu einer Navier-Stokes-Näherungslösung erster Ordnung in der Zeit und zweiter Ordnung im Raum ist.

Ein deutlicher Vorteil der Momentenmethode ist die Möglichkeit, die nicht-hydrodynamischen Relaxationsraten so einzustellen, dass Modellartefakte minimiert werden und dadurch die numerische Stabilität optimiert wird.

Die Benutzung der Momentenmethode verbessert die Effizienz substantiell. Die maximale Zellreynoldszahl ist bis zu zwei Größenordnungen größer gegenüber dem BGK-Ansatz.

2.1.3 Lattice Boltzmann Methode für Flachwasserströmungen

Szenarien wie Dammbruchsimulationen oder allgemein Simulationen freier Oberflächen sind auch für Laien einfach nachzuvollziehen. In der in dieser Arbeit entwickelten interaktiven Simulationsumgebung (siehe Kapitel 7) wird dies demonstriert. Hierfür wurde ein Lattice-Boltzmann-Modell für die Flachwassergleichungen implementiert. Dieses wurde von Zhou [143] entwickelt und kann akurate Lösungen für die Flachwassergleichungen berechnen. Adaptive Berechnungen wie in Kapitel 2.4.4 veranschaulicht, sind damit möglich. Zunächst werden die zugrundeliegenden Gleichungen gezeigt. Dem folgt ein einfaches Validierungsbeispiel der Strömung über eine Bodenwelle. Ausgangspunkt ist die allgemeine Form eines Lattice Boltzmann Modells zur Simulation von Strömungen:

$$f_i(t+\Delta t, \mathbf{x}+\mathbf{e}_i\Delta t) - f_i(t,\mathbf{x}) = -\frac{1}{\tau}(f_i - f_i^{eg}) + \frac{\Delta t}{6c^2}\mathbf{e}_i\mathbf{F} \tag{2.22}$$

$c = \frac{\Delta x}{\Delta t}$ ist die mikrokopische Referenzgeschwindigkeit und $\mathbf{F}$ die Kraft. Die kinematische Viskosität ν ist definiert mit:

$$\nu = \frac{c^2}{6}(2\tau - \Delta t) \tag{2.23}$$

Die lokalen Gleichgewichtsfunktionen sind:

$$f_i^{eq} = \begin{cases} h - \frac{5gh^2}{6c^2} - \frac{2h}{3c^2}\mathbf{u}^2, & i = 0 \\ \frac{gh^2}{6c^2} + \frac{h}{3c^2}\mathbf{e}_i\mathbf{u} + \frac{h}{2c^4}(\mathbf{e}_i\mathbf{u})^2 - \frac{h}{6c^2}\mathbf{u}^2, & i = E,W,N,S \\ \frac{gh^2}{24c^2} + \frac{h}{12c^2}\mathbf{e}_i\mathbf{u} + \frac{h}{8c^4}(\mathbf{e}_i\mathbf{u})^2 - \frac{h}{24c^2}\mathbf{u}^2, & i = NE,SW,SE,NW \end{cases} \tag{2.24}$$

Die makroskopischen Größen Wassertiefe h und Geschwindigkeit $\mathbf{u}$ sind folgendermassen mit den Verteilungen verknüpft

$$\sum_{i=0}^{b} f_i = h \qquad \sum_{i=0}^{b} \mathbf{e}_i f_i = h\mathbf{u} \tag{2.25}$$

Der Kraftterm wird zur Modellierung der Gravitationskraft, Windscherspannung, Corioliskraft [80] und des Bodengefälles verwendet. Zur Berechnung des Kraftterms wird ein zentrales Schema eingeführt [142]. Die Gleichung für den Kraftterm resultierend aus dem Bodengefälle ist:

$$\mathbf{F} = -gh\frac{\partial \mathbf{z}_b}{\partial \mathbf{x}} \tag{2.26}$$

$$f_i^F = \mathbf{e}_i\frac{\Delta t}{6c^2}(-g)\frac{h(\mathbf{x}) + h(\mathbf{x}+\mathbf{e}_i\Delta t)}{2}\frac{z(\mathbf{x}) - z(\mathbf{x}+\mathbf{e}_i\Delta t)}{\Delta x_i} \tag{2.27}$$

h und z sind die Wassertiefe und Bodenhöhe des betrachteten Knotens.

mit $\Delta x = \mathbf{e}_i\Delta t$ wird Gleichung 2.27 zu

$$f_i^F = -\frac{g}{12c^2}(h(\mathbf{x}) + h(\mathbf{x}+\mathbf{e}_i\Delta t))(z(\mathbf{x}) - z(\mathbf{x}+\mathbf{e}_i\Delta t)) \tag{2.28}$$

Strömung über eine Bodenwelle

In [143] beschreibt Zhou die Strömung über eine Bodenwelle. Durch diese erhöht sich die Geschwindigkeit der Strömung und somit entsteht an der Wasseroberfläche eine Absenkung. Hier wird der Testfall genutzt, um das implementierte Schema mit Gitterverfeinerung zu validieren.

Der Kanal ist $25m$ lang. Die Bodenhöhe z_b für die Bodenwelle wird folgendermaßen definiert:

$$z_b(x) = \begin{cases} 0.2 - 0.05(x-10)^2 & 8m < x < 12m \\ 0.0 & x \leq 8m, x \geq 12m \end{cases} \tag{2.29}$$

Am Einlauf ist ein Durchfluß von $q = 4.42m^2/s$ gesetzt. Der Auslauf hat eine Höhe von $h = 2m$. Die mikroskopische Referenzgeschwindigkeit ist $c = 15m/s$ und die Relaxationszeit $\tau = 1.5s$. Das Abbruchkriterium der Simulation ist eine Abweichung von 10^{-6} für die Änderung der Wasseroberfläche. Für die uniforme Berechnung wurde das Gebiet mit einem Gitter von 250×10 Knoten aufgelöst. Das entspricht einem Gitterabstand von $0.1m$. Das Ergebnis des zu erwartenden Tiefpunktes der Wasseroberfläche ist $1.90735m$ und die maximale Geschwindigkeit beträgt $2.5888\frac{m}{s}$. Der Durchfluß ist mit $q = 4.42m^2/s$ im Kanal konstant. Im Bereich von 7 bis 13 m wurde das Rechengitter stufenweise bis auf Level 3 verfeinert. Die Berechnungen mit Verfeinerung (siehe Abbildung 2.4) lieferten konsistente Ergebnisse, die mit den Sollwerten von [143] übereinstimmen.

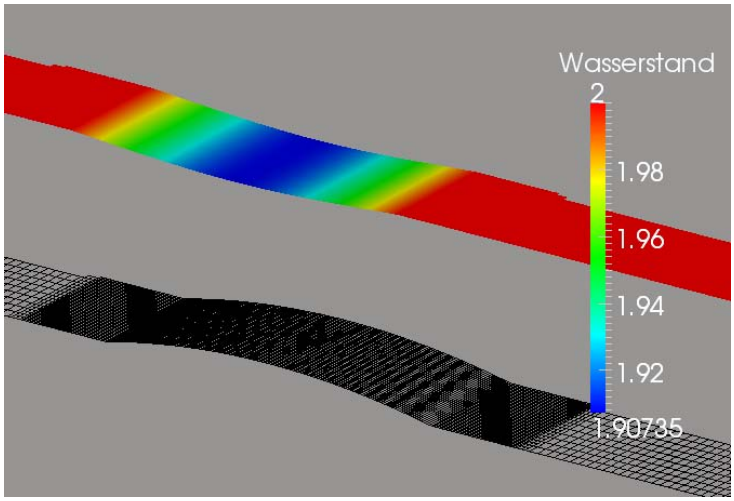


Abbildung 2.4: *Strömung über eine Bodenwelle mit Gitterverfeinerung*

2.2 Randbedingungen

Makroskopische Randbedingungen können nur implizit über die Partikelverteilungsfunktion gesetzt werden. Ein sehr verbreiteter und einfacher Weg zur Simulation einer Haftrandbedingung an Wänden ist das sogenannte Bounce-Back-Schema. Dabei werden die Verteilungen, die auf die Wand prallen, in entgegengesetzter Richtung reflektiert. Wenn die Wand einen halben Gitterabstand entlang des Gittervektors $\hat{\mathbf{e}}_i = \mathbf{e}_i/c$ entfernt ist, ist das Schema 2. Ordnung ansonsten nur 1. Ordnung. Da beliebig geformte und bewegte Körper betrachtet werden, wurde das modifizierte Bounce-Back-Schema, entwickelt in [7, 74], für Geschwindigkeitsrandbedingungen genutzt. Hierfür wird zusätzlich der Impulsaustausch zwischen Rand und Fluid berücksichtigt.

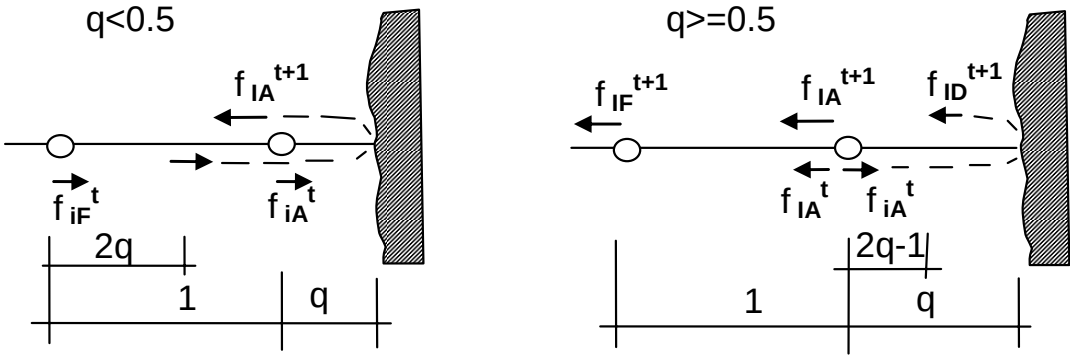


Abbildung 2.5: *Interpolationen für das modifizierte Bounce-Back-Schema*

In Abbildung 2.5 sind für das modifizierte Bounce-Back-Schema zwei Fälle gezeigt:

(a) normierter Wand–Knotenabstand $q_i < 0.5$

$$f_{IA}^{t+1} = (1-2q)f_{iF}^{t} + 2qf_{iA}^{t} - 2\rho w_i \frac{e_i u_w}{c_s^2} \quad mit\ 0.0 < q < 0.5 \tag{2.30}$$

(b) normierter Wand–Knotenabstand $q_i \geq 0.5$

$$f_{IA}^{t+1} = \frac{2q-1}{2q} f_{IA}^{t} + \frac{1}{2q} f_{iA}^{t} - \rho w_i \frac{e_i u_w}{q c_s^2} \quad mit\ 0.5 \leq q \leq 1.0 \tag{2.31}$$

u_w ist die Geschwindigkeit des entsprechenden Hindernisobjektes. Mit diesem Schema wird für gekrümmte Ränder eine Genauigkeit 2. Ordnung im Raum erhalten [37]. Für eine detailierte Betrachtung von LB-Randbedingungen wird auf [40, 41] verwiesen. Die dort entwickelten Multi-Reflection-Randbedingungen basieren auf Interpolationen höherer Ordnung und Korrekturtermen.

Im Gegensatz zum einfachen Bounce-Back ist diese Interpolationsbedingung nicht massenerhaltend. Dennoch sind die Ergebnisse mit dem einfachen Bounce-Back schlechter, was die Wichtigkeit einer ordnungsgemäßen geometrischen Auflösung des Strömungsgebietes hervorhebt.

Indem man die unbekannten Randverteilungen zu [128]

$$f_I(t+\Delta t, \mathbf{x}) = -f_i(t, \mathbf{x}) + f_I^{eq}\Big(p_0, \mathbf{u}(t_B, \mathbf{x}_B)\Big) + f_i^{eq}\Big(p_0, \mathbf{u}(t_B, \mathbf{x}_B)\Big) \tag{2.32}$$

setzt, werden Druckrandbedingungen gesetzt. p_0 ist dabei der gegebene Druck, $t_B = t + \frac{1}{2}\Delta t$, $\mathbf{x}_B = \mathbf{x} + \frac{1}{2}\mathbf{e}_i\Delta t$, (f_I, f_i) sind ein antiparalleles Paar mit den Geschwindigkeiten $\mathbf{e}_i = -\mathbf{e}_I$ und f_I ist der ins Gebiet kommende und f_i der aus dem Gebiet heraustretende Verteilungsfunktionswert. Die Geschwindigkeit $\mathbf{u}$ erhält man durch Extrapolation.

2.2.1 Aktivierung/Deaktivierung von neuen Fluidknoten

Wenn infolge der bewegten Struktur ein Knoten seinen Status von solid zu fluid ändert, werden die lokalen Geschwindigkeiten zu diesem Knoten, abhängig von der geometrischen Konfiguration, inter- bzw. extrapoliert. Eine lokale Poisson-artige Iteration, beschrieben in [85], wird benutzt um für diesen Knoten den Druck und die Momente höherer Ordnung zu berechnen.

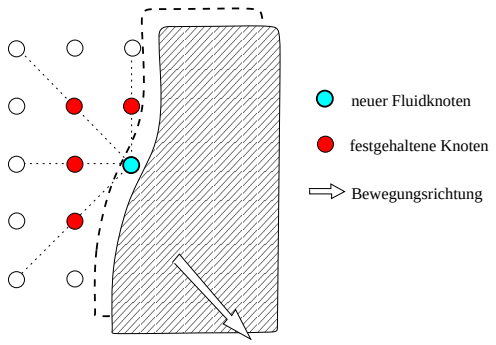


Abbildung 2.6: *Interpolation neuer Fluidknoten*

Dies bedeutet, dass die Lattice-Boltzmann-Gleichung solange lokal für den neuen Fluidknoten gelöst wird, bis dessen Verteilungen konvergiert sind. Die Verteilungen der Nachbarknoten bleiben konstant und wandern für jeden lokalen Schritt bei der Propagation zum neuen Knoten. Abbildung 2.7 und Abbildung 2.8 zeigen die internen Iterationen für einen fluidumströmten Kreis, der um den Abstand kleiner $\mathbf{e}_E\Delta t$ versetzt wurde.

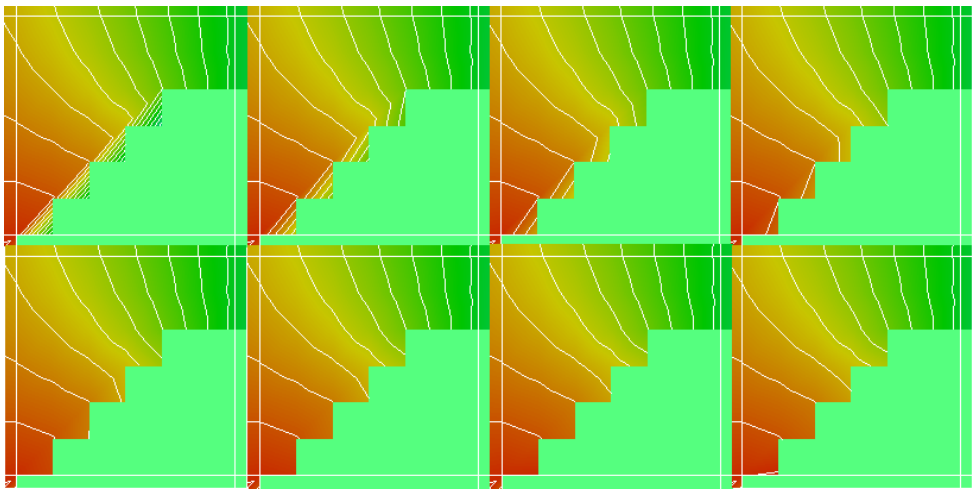


Abbildung 2.7: *interne Iteration für das Druckfeld*

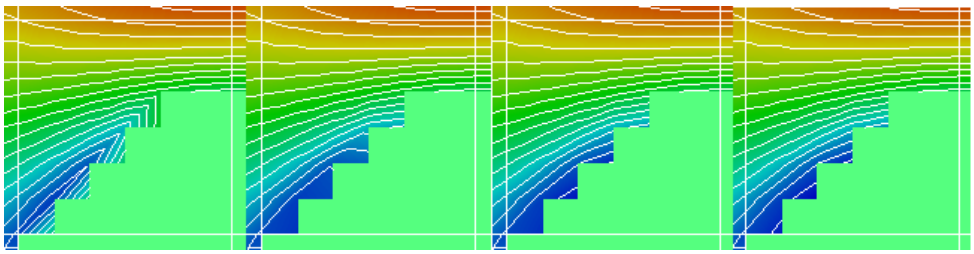


Abbildung 2.8: *interne Iteration für das Geschwindigkeitsfeld*

2.3 Gitterverfeinerung

Für eine effiziente Lösung von Computational Fluid Dynamics (CFD) Problemen sind lokal verfeinerte Gitter unabdingbar. Dies gilt vor allem für die Auflösung von Grenzschichten und Wirbelstraßen. Bei Simulationen mit großen Gradienten, wie z. B. dem Phasengradient bei Mehrphasensimulationen, werden die Gebiete mit hohem Gradienten hochaufgelöst gerechnet. Für Gebiete mit kleinem Gradienten genügt meist ein grobaufgelöstes Gitter. Der Hauptnutzen von lokal verfeinerten Gittern im Gegensatz zu vollaufgelösten Gittern ist die Rechenzeitersparnis bei vergleichbarer Genauigkeit.

Die LB-Methode benötigt als Basiszelle in zwei Raumdimensionen ein Quadrat und in drei Raumdimensionen einen Würfel. In der vorhergehenden Arbeit von Crouse [17, 18] wurden hierfür rekursiv verfeinerte Quadtree-Datenstrukturen in 2-D implementiert. Die „Quads“ - Blätter des Baumes haben hierbei die im LB-Kontext notwendigen Knoten an den Ecken gespeichert. Die Knoten werden in einem Feld von Hashtabellen [46] gespeichert, da diese im Gegensatz zu einer reinen Baumstruktur leichte Geschwindigkeitsvorteile bei der Auffindung von Nachbarzellen und bei adaptiv hinzukommenden bzw. wegfallenden Zellen haben. Die alternative Abspeicherung in Feldern benötigt zuviel Speicherplatz, da eine entsprechend große Matrix für jedes Gitterlevel angelegt werden muss.

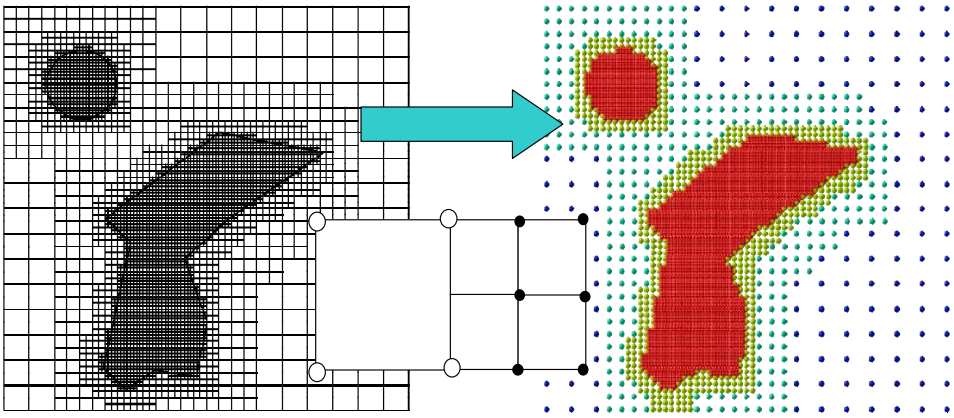


Abbildung 2.9: *Abbildung des Quadtrees auf eine Knotendatenstruktur*

Die Kopplung der Raum und Zeitdiskretisierung $\Delta x = c\Delta t$ führt zu einem gestaffelten Zeitschrittschema. Wenn die Schallgeschwindigkeit und die physikalische Viskosität auf allen Gitterleveln als gleich vorausgesetzt werden (Machzahl sowie Reynoldszahl gleich auf allen Gittern), so müssen zwei Zeitschritte auf dem feineren Gitter, während eines Zeitschrittes auf dem groben Gitter durchgeführt werden [29].

Rheinländer [102] wählte einen Ansatz bei dem die Machzahl auf den feineren Gitterleveln reduziert wird. Damit wird die Konvergenz zur inkompressiblen Navier-Stokes-Gleichung verbessert. Dieser Ansatz benötigt vier Zeitschritte auf dem feinen Gitter während eines Zeitschrittes auf dem groben Netz.

Die Abbildung 2.10 zeigt das gestaffelte Zeitschrittberechnungsschema für einen groben Zeitschritt. Der Levelindex ist mit der Gittergröße verknüpft, Level 0 entspricht Δx und Level 1 $\frac{1}{2}\Delta x$. Ein Zeitschritt in Level 0 bedeutet zwei Zeitschritte auf Level 1 und so weiter. Für eine absolute Anzahl k von Verfeinerungsstufen benötigt man 2^k interne Iterationen.

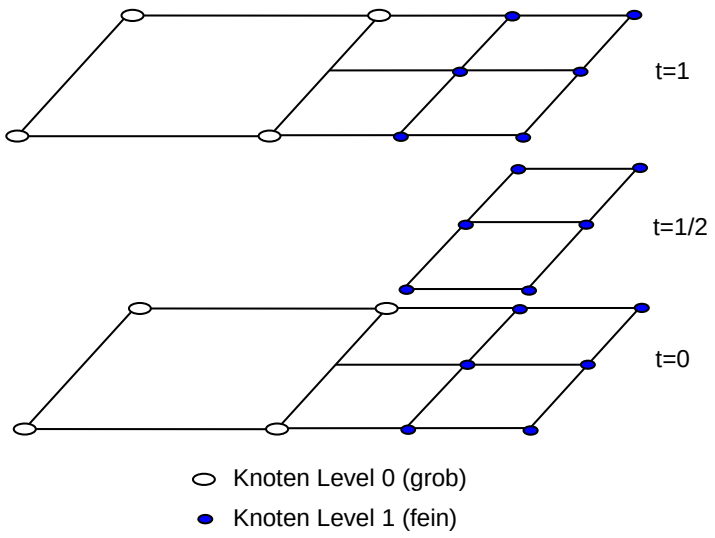


Abbildung 2.10: *Schematische Darstellung des Knotengitters in den verschiedenen Zeitebenen*

2.3.1 Theorie

Die Analyse der Lattice-Boltzmann-Gleichungen auf verschiedenen Gitterleveln zeigt, dass es nicht ausreicht, zwei kartesische Gitter verschiedener Auflösung durch eine Interpolation der Nichtgleichgewichtsanteile der Verteilungen während der Propagation vom groben zum feinen Gitter zu koppeln. Fillipova und Hänel haben erstmals in [29] gezeigt, dass man die Nichtgleichgewichtsverteilungen entsprechend skalieren muss, um störungsfreie Gitterübergänge für Druck, Geschwindigkeit und Dehnraten zu erhalten.

Die Herleitung der Skalierungsprozedur folgt hier analog der Veröffentlichung [140, 141]. Sie wird jedoch etwas modifiziert, da der Originalalgorithmus eine Singularität bei $\nu = 1/6$ aufweist.

Über dem Gitterinterface muss die Dichte und Geschwindigkeit auf dem feinen (f) und groben (g) Gitter gleich sein. Somit ergibt sich für die Gleichgewichtsverteilungen:

$$f^{g,eq} = f^{f,eq} \tag{2.33}$$

Der Index i wird der Übersicht halber weggelassen.

Die Nichtgleichgewichtsverteilungen $f^{neq} = f - f^{eq}$ werden durch die Gleichsetzung des deviatorischen Spannungstensors skaliert [17, 18, 29, 140, 141]:

$$\left(1 - \frac{\Delta t^g}{2\tau^g}\right) f^{g,neq} = \left(1 - \frac{\Delta t^f}{2\tau^f}\right) f^{f,neq} \tag{2.34}$$

Mit der Einführung des Skalierungsfaktors $sF^{f\to g}$ und dem Umstellen von Gleichung 2.34 ergibt sich:

$$f^{c,neq} = sF^{f\to g} f^{f,neq} = sF^{f\to g}(f^f - f^{f,eq}) \tag{2.35}$$

Substituiert man Gleichung 2.12 in Gleichung 2.34 folgt:

$$sF^{f\to g} = \frac{6\nu + \Delta t^g}{6\nu + \Delta t^f} \tag{2.36}$$

Gleichung 2.33 und Gleichung 2.35 eingesetzt in $f = f^{eq} + f^{neq}$ führt zu:

$$f^c = f^{f,eq} + sF^{f\to g}(f^f - f^{f,eq}) \tag{2.37}$$

Der Post-Kollisions-Status $\tilde{f}$ ist gegeben durch:

$$\tilde{f}^g = f^g - \frac{\Delta t^g}{\tau^g}(f^g - f^{g,eq}) \tag{2.38}$$

Gleichung 2.37 eingesetzt in Gleichung 2.38 ergibt:

$$\tilde{f}^g = f^{f,eq} + sF^{f\to g}(f^f - f^{f,eq}) - \frac{\Delta t^g}{\tau^g} sF^{f\to g}(f^f - f^{f,eq}) \tag{2.39}$$

resultierend in:

$$\tilde{f}^g = f^{f,eq} + \left(1 - \frac{\Delta t^g}{\tau^g}\right) sF^{f\to g}(f^f - f^{f,eq}) \tag{2.40}$$

Gleichung 2.40 berechnet den Kollisionschritt auf dem feinen Knoten und skaliert die Verteilungen zum Post-Kollisionszustand im groben Gebiet.

Für die Skalierung von grob nach fein ist der Skalierungsfaktor $sF^{g\to f} = \frac{1}{sF^{f\to g}}$.

Bei der Verwendung des MRT-Modells für die Gitterverfeinerung müssen nur die Nichtgleichgewichtsmomente $p_{xx}^{neq}, p_{xy}^{neq}$ so skaliert werden, dass die deviatorischen Spannungen am Gitterinterface kontinuierlich sind. Die Relaxationsraten für s_7 und s_8 sind durch Gleichung 2.12 auf allen Gitterleveln fixiert. Die anderen Relaxationsraten können für jedes Gitterlevel beliebig im Bereich $[0, 2]$ gewählt werden. Eine gute Wahl ist $s_1 = s_2 = s_4 = s_6 = 1$ auf allen Gitterleveln.

Die Beziehung zur Skalierung der Momente auf verschiedenen Gitterleveln nach dem Propagationsschritt lautet:

$$m^{g,neq} = \frac{s^f}{s^g}\frac{\Delta t^g}{\Delta t^f} m^{f,neq} \tag{2.41}$$

Diese Skalierung ist notwendig für einen kontinuierlichen Übergang der Momente über die Gitterlevel. Der Skalierungsfaktor beträgt zwei, wenn man man dieselbe Kollisionsrate auf dem groben und feinen Gitter nutzt.

2.3.2 Umsetzung

Im Gegensatz zum Standardansatz mit einem überlappenden Bereich beim Interface [18, 140] wird ein Gitterlayout ohne überlappende und hängende Knoten genutzt (vgl. Abbildung 2.9 und 2.11). Die Knoten sind in Hashtabellen gespeichert und haben zusätzliche Attribute, wie ihre Position und verschiedene Flags (boundary, scaleCoarseToFine, scaleFineToCoarse), womit Verteilungen, die skaliert oder Randbedingungen ausführen, markiert werden. Flags kennzeichnen die Richtungen, in die die Verteilungen nach dem Skalieren bzw. der Randbedingung propagiert werden. Somit kann man bei der Propagation über einfache Bitoperationen ermitteln, ob die Verteilungen normal oder skaliert verschoben werden bzw. über Randbedingungsalgorithmen berechnet werden. Die Bits zur Zerlegung eines Integerattributes für die acht Nicht-Null-Gitterrichtungen in 2-D sind folgendermaßen definiert: E=0x01, N=0x02, W=0x04, S=0x08, NE=0x10, NW=0x20, SW=0x40, SE=0x80. Beispielsweise ist der Wert eines Randbedingungsknotens, siehe Abbildung 2.11, 200 für die drei markierten Flags.

Sobald ein Knoten keinen Nachbarn hat, ist er entweder ein Randknoten oder ein Knoten, der eine skalierte Verteilung (von grob zu fein) erhält. Für solch einen Skalierungsknoten müssen die Verteilungen

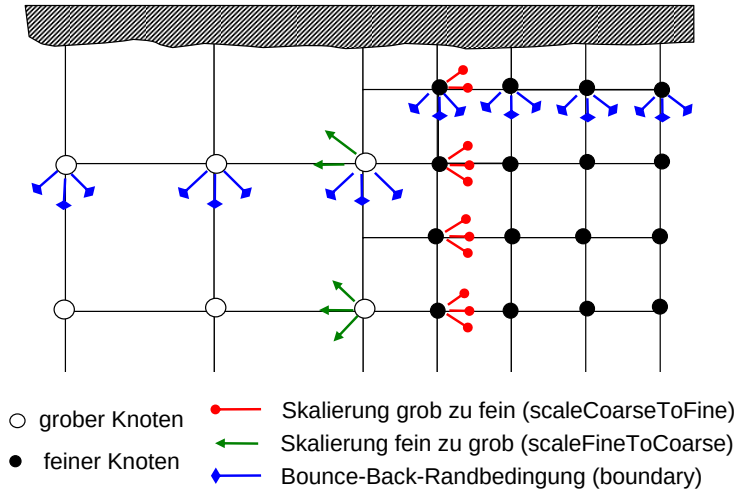


Abbildung 2.11: *Flags der Knoten*

mindestens quadratisch im Raum und linear in der Zeit interpoliert werden. Aufgrund der Symmetrie werden die Verteilungen kubisch im Raum interpoliert.

Ein Gitterknoten hat zwei Verteilungssätze, einen für den alten und einen für den neuen Zeitschritt (bezeichnet als f und f^{temp}).

Die Berechnung wird mittels gestaffelten Zeitschrittverfahrens mit zwei Iterationsschemen (a) dem *Forward-* und (b) dem *Straight-Iteration-Scheme* [11] durchgeführt. Abbildung 2.12 und Abbildung 2.13 zeigen die Iterationsschemen für ein verfeinertes Gitter bis zur Stufe 3. Die Startlevel für die acht internen Iterationen für einen groben Zeitschritt sind $0, 3, 2, 3, 1, 3, 2, 3$ für das *Forward-Iteration-Scheme* und $3, 2, 3, 1, 3, 2, 3, 0$ für das *Straight-Iteration-Scheme*.

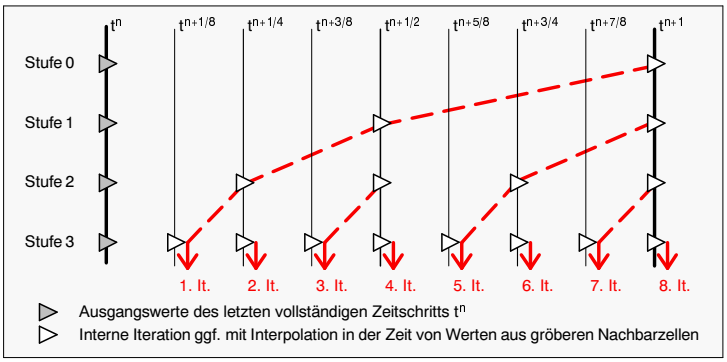


Abbildung 2.12: *Forward-Iteration-Scheme [11]*

Während sämtliche Berechnungsschleifen mit dem *Forward-Iteration-Scheme* arbeiten, verwendet die Methode, in der die temporären Verteilungen f^{temp} nach der Propagation zu den Verteilungen f gesetzt werden, das *Straight-Iteration-Scheme*.

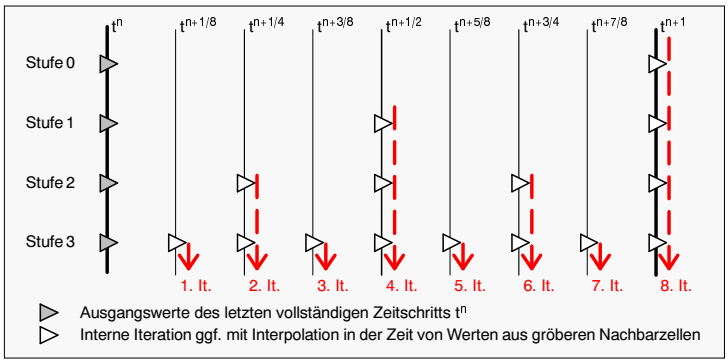


Abbildung 2.13: *Straight-Iteration-Scheme [11]*

Der generelle Algorithmus arbeitet wie folgt:

Algorithmus 2 gestaffeltes Zeitschrittverfahren

Starte Schleife über Zeitschritte gestaffelter Berechnung mit $2^{feinstes\ Level}$ internen Iterationen:

1. Berechne *Forward* Level und *Straight* Level für den internen Startlevel.
2. die skalierten Verteilungen werden in das f^{temp} Feld von Knoten mit Skalierungsflag propagiert.
   a) skaliere markierte scaleCoarseToFine Verteilungen für Knoten vom *Forward*-Level bis zum feinsten Level
   Die Verteilungen an dem feinen Knoten werden skaliert vom benachbarten groben Knoten mittels Gleichung 2.40. Feine Knoten ohne Nachbarn erhalten rauminterpolierte grobe Knotenverteilungen. Für die zeitliche Iteration, bei denen kein assoziierter Wert des groben Nachbarn vorhanden ist, werden diese durch eine lineare Zeitinterpolation erhalten.
   b) skaliere markierte scaleFineToCoarse Verteilungen für Knoten vom *Forward*-Level bis zum feinsten Level
3. Kollision für Knoten vom *Forward*-Level bis zum feinsten Level.
4. Propagation für Knoten vom *Forward*-Level zum feinsten Level.
   Jeder Knoten sammelt die Verteilungen von den benachbarten Knoten abhängig von den Randbedingungsregeln ein. Die Verteilungen werden vom f-Feld in das f^{temp}-Feld propagiert.
5. Setze f^{temp} Werte zu f bevor die nächste Iteration gestartet wird vom *Straight*-Level bis zum feinsten Level.

Mit zwei Feldern für die Verteilungen in der Datenstruktur gibt es zwei gültige Sätze für jeden Knoten nach der Propagation. Diese sind notwendig für die Kraftberechnung mit der Momentenaustauschmethode (siehe Kapitel 4.1) und sogenannte „Geisterknoten“ (Knoten, die im System redundant sind) können vermieden werden.

Zum besseren Verständnis wird der Algorithmus als Quelltext gezeigt:

```
int minInitLevel       = 0;
int maxInitLevel       = finestLevel-coarsestLevel;
int anzLevel           = maxInitLevel+1;
int straightStartLevel = minInitLevel;
int internalIterations = 1 << (maxInitLevel-minInitLevel);

for(int calcStep=1; calcStep<=anzCalcSteps; calcStep++)
{
  for(int staggeredStep=1; staggeredStep<=internalIterations; staggeredStep++)
  {
    forwardStartLevel = straightStartLevel;
    if(staggeredStep == internalIterations) straightStartLevel = minInitLevel;
    else
    {
       for(straightStartLevel=maxInitLevel,threshold=1;
          (staggeredStep&threshold)!=threshold; straightStartLevel--,threshold<<=1);
    }

    scale(forwardStartLevel, minInitLevel, maxInitLevel);
    collision(forwardStartLevel, maxInitLevel);
    propagate(forwardStartLevel, maxInitLevel);
    setFs(straightStartLevel, maxInitLevel);
  }
}
```

2.4 Gittergenerierung

Die Güte des Ergebnisses numerischer Verfahren hängt in erster Linie vom verwendeten Gebietsdiskretisierung ab. Gewöhnlich gilt: je feiner das Netz, desto besser die Lösung. Die Kehrseite der Medaille ist, dass die Berechnungsdauer meist polynomial mit der Anzahl der Freiheitsgrade ansteigt. Für LB-Simulationen bedeutet dies aufgrund der inheränten Kopplung der Physik mit dem numerischen Gitter in zwei Raumdimensionen einen kubischen Anstieg. Das bedeutet, dass es nicht ratsam ist, im ganzen Gebiet ein hoch aufgelöstes Gitter zu verwenden, sondern nur Bereiche zu verfeinern, in denen es notwendig ist. Dies führt zur Klassifikation von strukturierten und unstrukturierten Gittern. Bei einem strukturierten Gitter sind die Elemente auf eine Matrixstruktur abbildbar. Strukturierte Netze sind normalerweise leichter zu handhaben, da die Nachbarschaftstopologien konstant sind. Der einfachste Fall ist ein kartesisches Gitter. Unstrukturierte Netze sind durch die nicht konstante Anzahl von an einen Knoten angrenzenden Elementen definiert. Ein Vorteil ist, dass unstrukturierte Netze weniger Elemente haben und günstig für die Berechnung sind. Der Aufwand für die Verwaltung der Geometrie und die Implementierung hingegen steigt, da die Gebietstopologie nicht mit einer einfachen Matrixstruktur abgebildet werden kann.

In dieser Arbeit werden sogenannte Quadtrees in 2-D bzw. Octrees in 3-D für den Lattice-Boltzmann-Strömungslöser VIRTUALFLUIDS verwendet. Diese zählen zur Gruppe der strukturierten Netze, haben aber durch ihre Eigenschaft der rekursiven Unterteilung des Gebietes den Vorteil von unstruktrurierten Netzen (geringere Elementanzahl, Verfeinerung um Objektoberflächen).

Ein Quadtree ist eine reguläre Unterteilung eines Basisquadrats (Raster). Er wird z. B. zur Strukturierung von Rasterdaten benutzt und ist definiert durch eine schrittweise Vierteilung einer Basiszelle. Das bedeutet, dass eine Elternzelle vier Kindzellen hat und jede von diesen potentiell wiederum vier Kindzellen etc.

Bei der Gittergenerierung müssen zwei Aspekte betrachtet werden: Erstens die Generierung des quadtree/octree-artigen Gitters selbst und zweitens die automatische Berechnung der Abstände (q_i) für bewegte Strukturen, dargestellt in Abbildung 2.14. Zur Generierung des Gitters wurden im Laufe der Arbeit zwei Ansätze verfolgt. Ansatz 1 basiert auf der bereits in Kapitel 2.3 beschriebenen Knotendatenstruktur, in der die Knoten, d.h. die diskretisierten LB-Knoten selbst, pro Verfeinerungslevel in einer Hashtabelle gespeichert sind. Die Knotenstruktur kann adaptiv durch verschiedene Kriterien verfeinert bzw. vergröbert werden. Kriterien sind z. B. die Divergenz der Geschwindigkeit, die Wirbelstärke oder auch geometrische Bedingungen. Dieser Ansatz mit einer a priori Verfeinerung des Gebietes wurde von Freudiger und Hegewald [33] parallelisiert. Hierbei stellte sich heraus, das die Knotendatenstruktur zu speicherintensiv ist, um große Gebiete verteilt zu rechnen. Auch die Gebietszerlegung mit METIS [62] benötigt zuviel Arbeitsspeicher. Der Aufbau des Graphen benötigt genauso viel Speicher, wie das Rechengitter selbst. Detailierte Informationen hierzu sind in der Dissertation von Freudiger [32] beschrieben. Aufgrund dieser Mängel wurde Ansatz 2 - eine hybride Blockgitterdatenstruktur entwickelt. Das Gebiet wird hierbei mit Zellen beschrieben. Diese werden wiederum in einer quadtree/octree-artigen Struktur verwaltet, um Gebietsverfeinerungen zu ermöglichen. In den Zellen sind Matrizen mit den LB-Knoten enthalten. Die Anzahl der Knoten kann pro Richtung variieren, wichtig ist nur, dass sie untereinander einen uniformen Knotenabstand haben.

Die topologischen Informationen der Knoten bzw. Zellen werden in Matrixkoordinaten gespeichert. Es ist somit eine eindeutige Transformation von Knoten-/Zellkoordinaten zu realen Weltkoordinaten nötig. Für die Gittergenerierung gibt es somit zwei Möglichkeiten: Entweder transformiert man die Geometrien auf Matrixkoordinaten oder die Knoten/Zellen in reale Weltkoordinaten und bestimmt dann, ob Knoten/Zellen in der Geometrie verfeinert werden sollen.

2.4.1 Bestimmung der normalisierten Gitterabstände

Die Abbildung eines komplexen, bewegten geometrischen Objektes innerhalb des Fluidgebietes erfordert die Berechnung der normalisierten Gitterabstände (q_i) von Fluidknoten die am Rand des Festkörpers liegen. Zur Bestimmung von Solidknoten, die Knoten innerhalb des Strukturobjektes beschreiben, sind effiziente Punkt-in-Polyeder-Tests [43] kombiniert mit rekursiven Füllalgorithmen implementiert worden.

Für die Distanzbestimmung werden effiziente Raytracing-Algorithmen [1] pro diskreter Richtung $e_i \Delta t$ angewandt. Die Knoten/Zellen werden dabei mit dem Bounding-Box-Test vorsortiert.

Am Fluid-Struktur-Interface ist es notwendig, die Solidknoten und Transknoten (Knoten benachbart zum Rand des Hindernisobjektes) zu speichern. In der Nachbarschaft der Dreiecke des Hindernisses werden die minimalen Abstände (q_i) von den Fluidknoten berechnet. Hierfür wurden zwei Algorithmen implementiert. Während der erste alle Knoten innerhalb der Bounding-Box benutzt, verwendet der zweite die Knoten, die über eine Octree-Generierung bestimmt werden.

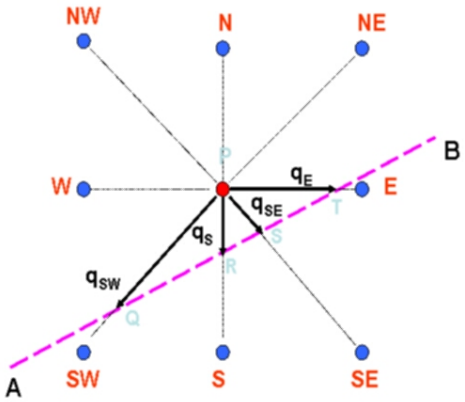


Abbildung 2.14: *Bestimmung der q_i in 2D*

Bounding-Box-basierte Methode

Es wird eine Bounding-Box erzeugt, die als minimale Koordinate den Startpunkt der Linie in 2-D und als maximale Koordinate den Endpunkt der Linie verwendet. Die Bounding-Box wird nun in jede Richtung um den Knotenabstand $\mathbf{e}_i \Delta t$ vergrößert. Für die Berechnung der q_i werden nur die Knoten betrachtet, die sich in der Bounding-Box befinden und die auf der richtigen Seite der Halbebene liegen (positiver Normalenvektor der Linie).

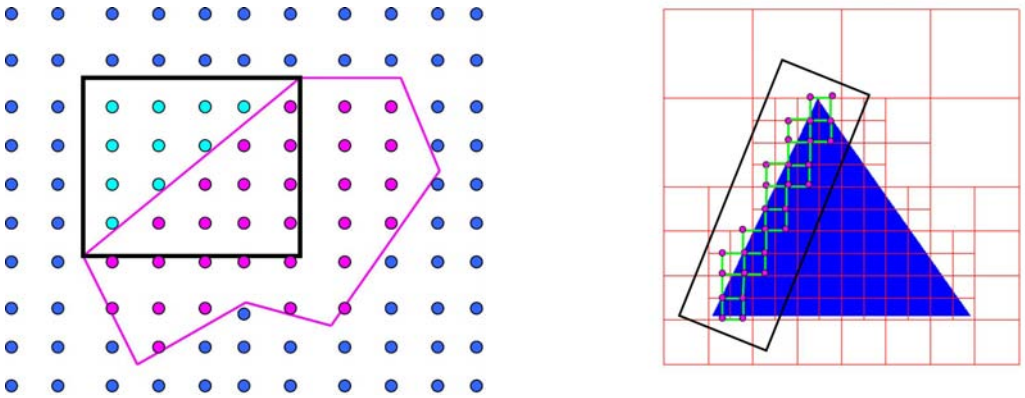
Jeder Knoten hat gemäß dem D2Q9-Modell in den acht Raumrichtungen Nachbarn. Es wird für jede Richtung überprüft ob die Linie die Kante schneidet.

Für den Fall, dass ein Schnittpunkt auftritt (siehe Abbildung 2.14), werden die Längen PQ, PR, PS und PT bestimmt, normalisiert und als q_i für die entsprechende Richtung hier, q_{SW}, q_S, q_{SE} und q_E abgespeichert.

Für den 3-D Fall ist das Basiskonzept identisch. Hier werden Dreiecke statt Linien betrachtet. Für das D3Q19-Modell werden 18 Nachbarrichtungen berücksichtigt.

Octree-basierte Methode

Die Octree-basierte Methode läßt sich durch die Betrachtung eines 2-D Falles verständlicher beschrieben. Der Unterschied zwischen diesem Algorithmus und dem Bounding-Box-Algorithmus ist, dass hier



(a) (b)

Abbildung 2.15: *Bounding Box und Octree basierte Methode*

Zellen und nicht Knoten betrachtet werden. Zuerst erhält man eine erzeugte Quadtreestruktur, bei der die feinsten Zellen entlang der zu betrachtenden Kante liegen. Die Zellen haben die Größe des feinsten Gitterlevels des Knotengitters. In 3-D erhalten wir eine Octreestruktur entlang des Dreiecks. Es werden nur die Eckknoten der feinsten Zellen betrachtet und die Abstände (q_i) bestimmt. Abbildung 2.15b zeigt die Knoten, die durch den Quadtree/Octree bestimmt werden.

Es hat sich herausgestellt, dass die Bounding-Box-basierte Methode einen signifikanten Laufzeitvorteil gegenüber der Octree-basierten Methode bei der Fluid-Struktur-Interaktion, wo die Objektoberflächen mit kleinflächigen Dreiecken beschrieben werden, hat. Der Vorteil der Octree-basierten Methode kommt erst bei großflächigen Dreiecksnetzen zum Vorschein.

2.4.2 Punkt-in-Polyeder-Test

Für bewegte Objekte in 3-D ist ein effizienter Punkt-in-Polyeder-Test notwendig, um zu bestimmen, ob ein Punkt innerhalb oder außerhalb des Polyeders liegt. Hier wurden im Rahmen einer Masterarbeit [63] drei verschiedene Algorithmen implementiert. Der Erste ist der Halbebenen-Test, der Zweite bildet die Dreiecke auf einer Kugel ab und der Dritte ermittelt mittels Raytracing die Anzahl der Schnittpunkte.

Halbebenen-Algorithmus

Der am einfachsten zu implementierende Algorithmus ist der Halbebenen-Algorithmus [6]. Dieser ist nur gültig bei konvexen Polyedern. Konvexe Polyeder zeichnen sich dadurch aus, dass jede beliebige Kante zwischen zwei Punkten des Polyeders vollständig innerhalb des Körpers liegt und keine weiteren Schnittpunkte mit diesem aufweist. Hierbei ist es wichtig, dass die Normalen der Facetten des Polyeders nach außen zeigen. Ein beliebiger Punkt gilt als innerhalb, wenn er in negativer Normalenrichtung aller Facetten liegt. Die Normale jeder Facette kann verhältnismäßig einfach bestimmt werden und sie zeigt nach außen. Alle Punkte, die entgegengesetzt der Normalenrichtung liegen, werden markiert. Wenn diese Berechnung auf allen Facetten ausgeführt wird, kann bestimmt werden, ob ein Punkt innerhalb oder außerhalb des konvexen Polyeders liegt. Die Beschränkung auf konvexe Körper ist ein großer Nachteil dieses Algorithmus. Es existieren jedoch Algorithmen, die jeden nicht-konvexen Polyeder in eine endliche Menge konvexer Polyeder zerlegen, die jeweils mit dem Halbebenen-Algorithmus analysiert werden können.

Solid-Angle-Algorithmus

In diesem Abschnitt wird der Solid-Angle-Ansatz für den Punkt-in-Polyeder-Test von Carvalho and Cavalcanti [12] beschrieben. Dieser Ansatz ist eine Erweiterung des zweidimensionalen Ansatzes von Haines [43]. Zuerst wird das Problem in 2-D beschrieben. Man kann durch die Berechnung des mit Vorzeichen versehenen Winkels um einen Punkt p für jede Kante des Polygons P bestimmen, ob p innerhalb oder außerhalb eines Polygons liegt. Für den Fall, dass p nicht auf dem Rand von P liegt, ist die Summe von allen Winkeln -2π, 0 oder 2π. Wenn die Summe 0 ist, liegt der Punkt außerhalb, ansonsten innerhalb des Polygons. Die Erweiterung der Methode auf 3-D [12] wird nachfolgend diskutiert.

In 2-D wird die Winkelabmessung bezogen auf eine Kante berechnet. Das ist das Maß eines gerichteten Bogens, der durch die Projektion der Kante auf einen Einheitszirkel, dessen Mittelpunkt der zu testende Punkt ist, erhalten wird. Der Bogen ist positiv, wenn seine Orientierung entgegen dem Uhrzeigersinn liegt. In Abbildung 2.16 zeigt der Winkel θ_{AB} den Winkel für die Kante AB. Hierfür erhält man den dazugehörigen Bogen $A'B'$ auf dem Kreis. Die entsprechende Operation in 3-D projeziert jedes Dreieck eines Polyeders auf eine Einheitskugel, wie man in Abbildung 2.17 sehen kann.

Als nächstes wird die Fläche des sphärischen Polygons (der Dreiecksfacette) bestimmt. Die Summe der projezierten Flächen von allen Facetten des Polyeders auf die Einheitskugel mit dem Mittelpunkt p ist 0, 4π oder -4π. Wenn die Summe 0 ist, liegt der Punkt außerhalb, ansonsten liegt er innerhalb. Die Fläche des sphärischen Polygons (Dreiecks) wird durch die klassische Formel von Girard bestimmt. Das Theorem von Girard besagt:

Die Fläche eines sphärischen Dreiecks auf einer Einheitskugel ist gleich der Summe der Dreieckswinkel minus zwei rechte Winkel. Die Fläche beträgt $S = A + B + C - \pi$, wobei A, B und C die sphärischen Winkel der Dreiecksknoten sind.

Der Solid-Angle-Test ist ein relativ ineffizienter Test zur Bestimmung, ob ein Punkt innerhalb oder außerhalb eines Polyeders liegt. Ein Grund dafür ist, dass die Winkelberechnung viele Operationen benötigt und der Test somit langsam für komplexe Fälle ist. Ein anderer Nachteil des Algorithmus ist das Fehlen eines Testes, ob ein Punkt auf dem Rand der Dreiecke des Polyeders liegt. Daher werden Punkte auf dem Rand nicht richtig bestimmt.

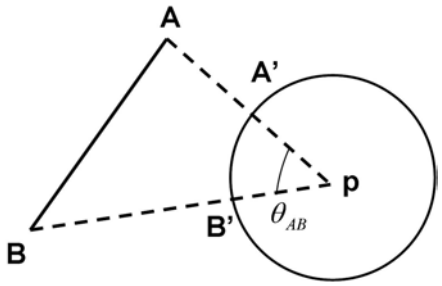


Abbildung 2.16: *Solid-Angle-Methode in 2-D, Kante $\overline{AB}$ eines Polygons*

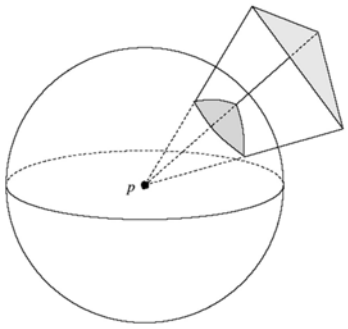


Abbildung 2.17: *Solid-Angle-Methode in 3-D [12], Facette eines Polyeders*

Ray-Crossing-Algorithmus

Der Ray-Crossing-Algorithmus vorgestellt von O'Rourke [97, S. 245-252] ist ein sehr effizienter Algorithmus zur Bestimmung, ob sich ein Punkt inner- oder außerhalb eines Polyeders befindet. Hierzu wird ein Strahl (ray) ausgehend vom zu überprüfenden Punkt bis zu einem Punkt außerhalb des Gebietes erzeugt und die Anzahl der Schnittpunkte des Strahls mit dem Polyeder bestimmt. Ein Punkt ist innerhalb des Körpers, wenn die Anzahl an Schnittpunkten ungerade ist, andernfalls liegt er außerhalb.

Auch wenn die Regel verhältnismäßig einfach klingt, ist eine Aussage des Ray-Crossing-Algoritmus nicht immer eindeutig. Dies liegt an den verschiedenen Positionen, an denen sich ein Punkt befindet, wie z. B. auf der Fläche einer Facette oder auf der Kante. Es kommt auch zu Unstimmigkeiten, wenn der Strahl eine Kante des Polyeders schneidet. Hier ergeben sich für den Schnittpunkttest mehrere Schnittpunkte und die Aussage, dass mit ungerader Anzahl von Schnittpunkten der Punkt innerhalb der Geometrie liegt, gilt nicht mehr zuverlässig.

Der Ray-Crossing-Algorithmus arbeitet folgendermaßen: Zuerst wird ein zufälliger Strahl generiert, der von einem beliebigen Punkt außerhalb des zu untersuchenden Körpers in Richtung des zu betrachtenden Punktes geht. Dann wird die Anzahl der Schnittpunkte bestimmt. Für den Fall, dass eine Uneindeutigkeit auftaucht, wird ein neuer Zufallsstrahl gewählt. Bevor für jede Facette der Schnittpunkt mit dem Strahl berechnet wird, führt man einen Bounding-Box-Test für den gesamten Polyeder durch. Bei der Schleife über alle Dreiecke wird ebenfalls erst gegen die jeweilige Bounding-Box geprüft.

Füllalgorithmus

Einer der bekanntesten Füllalgorithmen ist der Flood-Fill-Algorithmus. Zuerst wird ein Punkt innerhalb des Polyeders mit einem der oben beschriebenen Tests lokalisiert. Danach betrachtet man die Nachbarn in allen 18 Gitterrichtungen in 3-D und markiert die Knoten als Solid solange rekursiv, bis ein Randknoten erreicht ist. Auf diesem Weg wird der Polyeder mit Solidknoten aufgefüllt, wobei nur ein Knoten als innenliegend identifiziert werden muß.

2.4.3 Dreiecksoberflächennetze der Struktur

Komplexe Geometrien in 3-D werden durch triangulierte Oberflächennetze, die im STL (Stereolithografie) Dateiformat [118] vorliegen, direkt aus CAD-Daten erzeugt. Bei der Fluid-Struktur-Kopplung mit dem Finite-Element-Programm ADHOC wird ein trianguliertes Austauschnetz vorgegeben. Man kann jede Oberfläche mittels triangulierten Oberflächen approximieren. Zur Verfeinerung des Fluidgebietes wurden Algorithmen implementiert, bei denen man für jedes Dreieck die vergrößerte Bounding-Box bestimmt, alle Knoten darin auf den Abstand zum Dreieck testet und den Knoten gegebenfalls verfeinert.

Betrachtet man nur den Abstand in Normalenrichtung zum Dreieck, ergeben sich bei stark gekrümmten Flächen Übergänge, an denen keine Verfeinerung stattfindet. Hier ist es angebracht, alle Knoten in der vergrößerten Bounding-Box zu verfeinern oder einen Prismenkörper auf dem Dreieck, in dem verfeinert werden soll, zu definieren. Im Code, der auf der Blockdatenstruktur beruht, werden hier die einzelnen Blöcke vergrößert und getestet ob das Dreieck den Block schneidet. Somit ergibt sich hier nicht der Nachteil der stark gekrümmten Flächen.

Es folgt die Beschreibung des Algorithmus zur Bestimmung der normalisierten Knotenabstände (q_i) von Dreiecksnetzen für den Blockgittercode.

Algorithmus 3 Bestimmung der normalisierten Knotenabstände

(1) für jede Bounding-Box eines Dreiecks des Netzes werden die Blöcke des Blockgitters ermittelt

(2) mittels eines Dreieck/Block Verschneidungstests werden weitere nicht relevante Blöcke aussortiert

(3) jeder Knoten dieser Blöcke wird gegen die Bounding-Box des Dreiecks getestet

(4) Knoten die innerhalb der Bounding-Box und innerhalb des Dreiecksnetzes liegen, werden mittels Halbebenentest aussortiert

(5) für die restlichen Knoten erfolgt die q-Bestimmung mittels effizienter Raytracing-Algorithmen für die diskreten Boltzmannrichtungen

Zur Markierung der nicht aktiven Blöcke (Blöcke innerhalb der Geometrie) und von Solidknoten wurden die Blöcke in der Bounding-Box markiert. Für nicht markierte Blöcke genügt ein Punkt-in-Objekt-Test, um den gesamten Block bei Erfolg als nicht relevant zu markieren. Für markierte Blöcke wird ein Solidpunkt bestimmt und ein rekursiver Füllalgorithmus durchgeführt.

2.4.4 Adaptive Gitterverfeinerung

Im Hinblick der Erhöhung der Genauigkeit des LB-Lösers wurde auch eine Funktionalität zur adaptiven Gitterverfeinerung implementiert. Die Grundidee ist, ein hochaufgelöstes Netz im Bereich des Randes des Hindernisobjektes zu erstellen sowie Bereiche, in denen bestimmte Kriterien zutreffen, zu verfeinern bzw. zu vergröbern. In Abbildung 2.18 sieht man eine geometrische Verfeinerung um eine Kugel eines Blockgitters. Die weißen Linien auf der Kugel entsprechen den normalisierten Knotenabständen (q_i). Ein weiteres Beispiel ist in Abbildung 2.19 gezeigt. Hier ist das Ergebnis einer Flachwassersimulation, die mit dem LB-Verfahren [143] berechnet wurde, gezeigt. Das Netz wird anhand eines Schwellwertes für den Gradienten der Wellenfront verfeinert. In Regionen, in denen der Gradient sehr klein ist, wird das Netz vergröbert. Es handelt sich hierbei um eine gitteradaptive Berechnung.

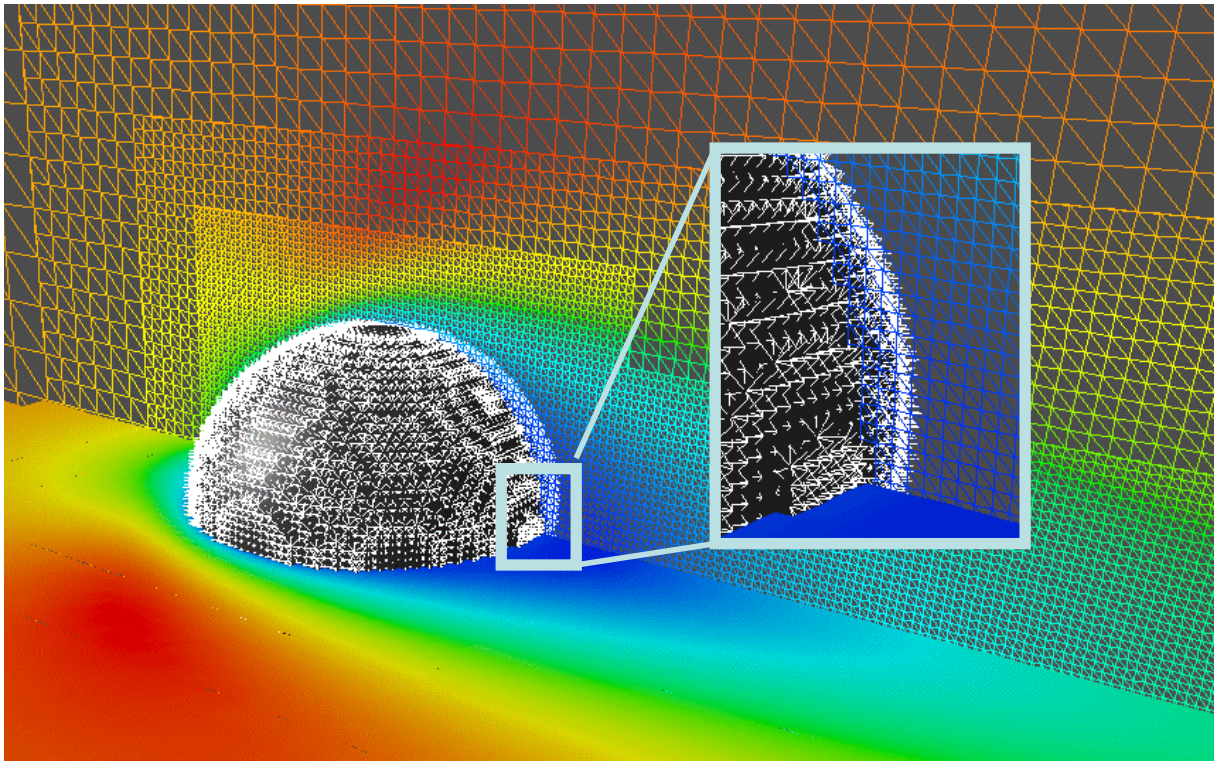


Abbildung 2.18: *a priori verfeinertes Blockgitter um eine Kugel*

Zur geometrischen Verfeinerung existieren mehrere Ansätze. Ein einfacher Ansatz ist, innerhalb eines vordefinierten geometrischen Objektes zu verfeinern. Hierbei werden alle Knoten überprüft, ob sie innerhalb der Geometrie sind und gegebenfalls verfeinert. Einen Knoten in 3-D zu verfeinern bedeutet, dass man ihn durch 27 neue Knoten ersetzt, falls keine benachbarten feinen Knoten vorhanden sind. Das Layout des zugrundeliegenden Gitters wird exemplarisch für zwei Raumdimensionen in Abbildung 7.1 in Kapitel 7.1 gezeigt, wobei neun neue Knoten bei der Verfeinerung gesetzt werden. Bei starren Hindernisgeometrien benötigt man keine Verfeinerung innerhalb des Objektes. Hier werden zwei Geometrien erzeugt, die eine größer mit definierten Abstand, die andere etwas kleiner als die Ursprungsgeometrie. Alle Knoten, die sich zwischen den beiden Oberflächen befinden, werden verfeinert. Dies funktioniert sehr gut für primitive Geometrien wie Kugeln oder Würfel. Umgesetzt wurde dies mit sogenannten Adaptionskriterien, die in Kapitel 7 softwaretechnisch beschrieben werden. Ein Nachteil ist hier, dass alle Knoten des Berechnungsgebietes getestet werden, was bei komplexen Geometrien, wie z. B. Polyedern mit vielen Dreiecken, in einem hohen Berechnungsaufwand resultiert.

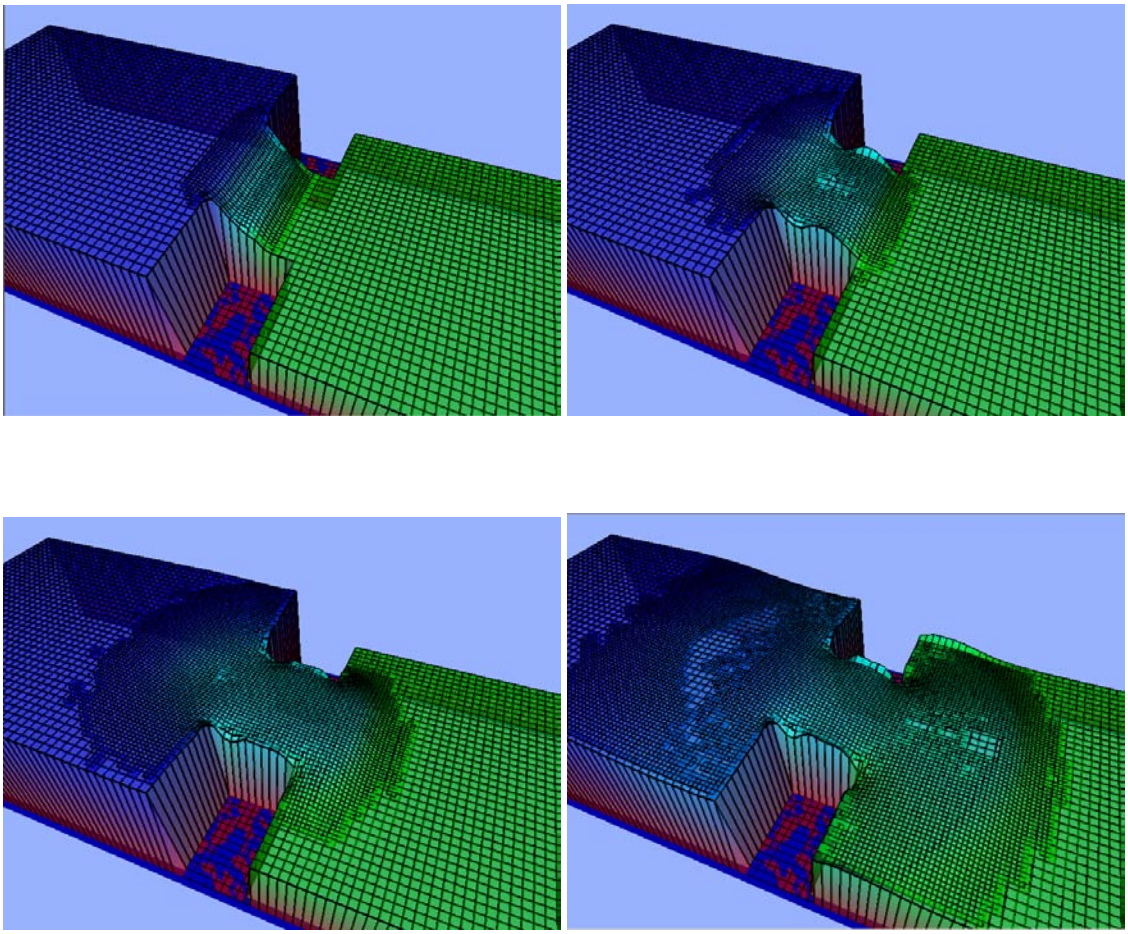


Abbildung 2.19: *adaptive Flachwassersimulation eines Dammbruches*

3 Finite Element Methode für die Strukturdynamikanalyse

Der LB-Löser wurde im Laufe dieser Arbeit mit zwei Strukturlösern gekoppelt. Der Erste ist ein zeitabhängiger Strukturlöser basierend auf eindimensionalen Balkenelementen. Dieser wurde implementiert, um die Fluid-Struktur-Kopplung zu entwickeln, zu testen und zu validieren. Dessen Modell ist ein Euler-Bernoulli-Balken und die Finite-Element-Diskretisierung basiert auf Hermite-Elementen mit Knotenverschiebungen und Verdrehungen als unabhängige Freiheitsgrade [86]. Für die Zeitintegration wurde das Newmarkschema implementiert. Alternativ wurde die Strukturlösung mit einem Eigenwertlöser berechnet.

Der zweite Löser ist der dreidimensionale transiente Strukturlöser ADHOC [22]. Er basiert auf Finiten Elementen hoher Ordnung (p-version) für die Raumdiskretisierung. Die Zeit wird wahlweise durch einen Newmarkansatz [93] oder die Generalized-α-Methode [16] diskretisiert. Außerdem können die Eigenfrequenzen und die auf die Masse normierten Eigenvektoren berechnet werden, wodurch die Lösung für geometrisch lineare Strukturen auch mittels Modalanalyse möglich ist.

In den folgenden Abschnitten werden die zugrundeliegenden Strukturgleichungen sowie die Vorteile bei der Simulation mit Finiten Elementen hoher Ordnung beschrieben.

3.1 Grundgleichungen der Strukturdynamik basierend auf der Elastizitätstheorie

3.1.1 Allgemeine Grundlagen [98, 144]

Die Gleichgewichtsbedingungen für die drei Raumrichtungen werden an einem differentiellen Kontinuumselement aufgestellt:

$$\begin{aligned}
\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{21}}{\partial x_2} + \frac{\partial \sigma_{31}}{\partial x_3} + p_{v1} = 0 \\
\frac{\partial \sigma_{12}}{\partial x_1} + \frac{\partial \sigma_{22}}{\partial x_2} + \frac{\partial \sigma_{32}}{\partial x_3} + p_{v2} = 0 \\
\frac{\partial \sigma_{13}}{\partial x_1} + \frac{\partial \sigma_{23}}{\partial x_2} + \frac{\partial \sigma_{33}}{\partial x_3} + p_{v3} = 0
\end{aligned} \tag{3.1}$$

σ ist die Spannung und p_v die Volumenlast, die auf den belasteten Körper wirkt. Wenn man nicht an Starrkörperbewegungen interessiert ist und sich das differentielle Element im Gleichgewicht befindet und somit nicht rotiert oder translatiert, fordert das Momentengleichgewicht, dass die Summe aller Momente gleich Null ist. Es gilt:

$$\sigma_{12} = \sigma_{21}\,,\ \sigma_{13} = \sigma_{31}\,,\ \sigma_{23} = \sigma_{32} \tag{3.2}$$

Die kinematischen Beziehungen verknüpfen die Dehnung ϵ_{ij} mit den Verschiebungen u_i:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \tag{3.3}$$

Zwischen den einzelnen Dehnungen müssen die Verträglichkeitsbedingungen (Kompatibilitätsbedingungen von Saint-Venant) bestehen, damit es möglich ist aus den sechs Dehnungen die

drei Verschiebungen eindeutig zu ermitteln.

$$\begin{aligned}
&\frac{\partial^2\epsilon_{11}}{\partial x_2^2}+\frac{\partial^2\epsilon_{22}}{\partial x_1^2}=2\frac{\partial^2\epsilon_{12}}{\partial x_1\partial x_2}\\
&\frac{\partial^2\epsilon_{22}}{\partial x_3^2}+\frac{\partial^2\epsilon_{33}}{\partial x_2^2}=2\frac{\partial^2\epsilon_{23}}{\partial x_2\partial x_3}\\
&\frac{\partial^2\epsilon_{33}}{\partial x_1^2}+\frac{\partial^2\epsilon_{11}}{\partial x_3^2}=2\frac{\partial^2\epsilon_{31}}{\partial x_3\partial x_1}\\
&\frac{\partial}{\partial x_1}\left(-\frac{\partial\epsilon_{23}}{\partial x_1}+\frac{\partial\epsilon_{13}}{\partial x_2}+\frac{\partial\epsilon_{12}}{\partial x_3}\right)=\frac{\partial^2\epsilon_{11}}{\partial x_2\partial x_3}\\
&\frac{\partial}{\partial x_2}\left(\frac{\partial\epsilon_{23}}{\partial x_1}-\frac{\partial\epsilon_{31}}{\partial x_2}+\frac{\partial\epsilon_{21}}{\partial x_3}\right)=\frac{\partial^2\epsilon_{22}}{\partial x_3\partial x_1}\\
&\frac{\partial}{\partial x_3}\left(\frac{\partial\epsilon_{32}}{\partial x_1}+\frac{\partial\epsilon_{31}}{\partial x_2}-\frac{\partial\epsilon_{12}}{\partial x_3}\right)=\frac{\partial^2\epsilon_{33}}{\partial x_1\partial x_2}
\end{aligned} \tag{3.4}$$

Somit lässt sich Gleichung (3.3) mit Hilfe der Differentialmatrix

$$\mathsf{D}^T=\begin{bmatrix}
\frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3}\\
0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_3} & 0\\
0 & 0 & \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1}
\end{bmatrix} \tag{3.5}$$

in folgende Schreibweise überführen:

$$\boldsymbol{\epsilon}=\mathsf{D}\mathbf{u} \tag{3.6}$$

Die letzte zur Beschreibung des Kontinuums noch fehlende Grundgleichung ist das Materialgesetz, das die Spannungen mit den Dehnungen verbindet. Der Zusammenhang beider Größen ist materialabhängig und wird experimentell ermittelt. In dieser Darstellung wird von einem linearelastischen Materialverhalten ausgegangen. Zusätzlich wird der Werkstoff als homogen und isotrop angenommen. Somit gilt für ideal-elastisches Material das Hookesche Gesetz:

$$\begin{aligned}
\sigma_{11}&=\frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\epsilon_{11}+\nu(\epsilon_{22}+\epsilon_{33})\right)\\
\sigma_{22}&=\frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\epsilon_{22}+\nu(\epsilon_{33}+\epsilon_{11})\right)\\
\sigma_{33}&=\frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\epsilon_{33}+\nu(\epsilon_{11}+\epsilon_{22})\right)\\
\sigma_{12}&=\frac{E}{(1+\nu)}\epsilon_{12}\\
\sigma_{13}&=\frac{E}{(1+\nu)}\epsilon_{13}\\
\sigma_{23}&=\frac{E}{(1+\nu)}\epsilon_{23}
\end{aligned} \tag{3.7}$$

Mit Hilfe der Elastizitätsmatrix

$$\mathsf{E} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{bmatrix} \tag{3.8}$$

lässt sich das Materialgesetz wie folgt beschreiben:

$$\boldsymbol{\sigma} = \mathsf{E}\boldsymbol{\epsilon} \tag{3.9}$$

Das Elastizitätsmodul E und die Querkontraktionszahl ν sind unter Berücksichtigung der physikalischen Grenzen $0 < E < \infty$ und $0 < \nu < 0.5$ für jedes Material experimentell bestimmbar.

3.1.2 Integralform der Bestimmungsgleichungen

Grundlage für die Methode der Finiten Elemente ist die Arbeitsgleichung 3.10, welche auch schwache Form der Gleichgewichtsbedingungen genannt wird. Grundlage für diese schwache Form sind die als starke Form bezeichneten Differenzialgleichungen die im vorherigen Abschnitt vorgestellt wurden. Diese werden zunächst mit einer geeigneten Wichtungsfunktion multipliziert. Als Wichtungsfunktionen werden in dem hier eingesetzten Galerkin Verfahren dieselben Funktionen gewählt, mit denen die unbekannten Feldgrößen angesetzt werden. Durch partielle Integration werden sodann die Anforderungen an die Differenzierbarkeit der Ansätze erniedrigt. Schreibt man alle Terme auf die rechte Seite, so erhält man ein Residuum. Dieses beschreibt das Gleichgewicht zwischen den äußeren und den inneren Kräften und ist nur im integralen Sinne erfüllt.

$$\int_\Omega \delta\boldsymbol{\epsilon}^T : \boldsymbol{\sigma} d\Omega - \int_\Omega \delta\mathbf{u}^T \mathbf{p}_\Omega d\Omega - \int_{\Gamma_p} \delta\mathbf{u}^T \mathbf{p}_\Gamma d\Gamma = \mathbf{0} \tag{3.10}$$

Der erste Term auf der linke Seite stellt die (virtuelle) Arbeit der inneren Spannungen dar. Der zweite Term beschreibt die (virtuelle) Arbeit aus Volumenlasten, wobei $\mathbf{p}_\Omega$ die Volumenlast repräsentiert. Sie wirkt im Innern des Gebietes Ω. Die eingeprägten Randspannungen werden mit $\mathbf{p}_\Gamma$ symbolisiert. Sie wirken auf der Teiloberfläche Γ_p und bilden mit den arbeitskonjungierten (virtuellen) Verschiebungen $\delta\mathbf{u}$ die virtuelle Arbeit aus äußeren Lasten.

3.2 Approximation mit finiten Elementen

Mit Hilfe der Finite Element Methode kann die Integralgleichung (3.10) diskretisiert werden. Dazu muss das zu untersuchende Gebiet vorab in geometrische Elemente endlicher Größe unterteilt werden, was zu einer Aufteilung des Integrals in eine endliche Menge von Teilintegralen über jedes einzelne Element führt. Aufsummiert ergeben diese Teilintegrale wieder das Ausgangsintegral. Jedes Teilintegral wird mit Hilfe normierter Koordinaten berechnet. Dabei werden die Zustandsgrößen $\boldsymbol{u}$ elementweise approximiert.

Das Verschiebungsfeld u wird auf Elementebene mit m Ansatzfunktionen S^i und Knotenweggrößen diskretisiert. Jede Knotenweggröße u^i ist einer bestimmten Ansatzfunktion S^i zugeordnet. Zusammen beschreiben sie den i-ten Freiheitsgrad des Elementes, so dass die Annäherungslösung $\tilde{u}$ für die Verschiebung eines einzelnen Elementes als Superposition von m Ansätzen geschrieben werden kann:

$$\tilde{\mathbf{u}}_e = \sum_{i=1}^{m} S^i(\eta_1, \eta_2, \eta_3) u^i \tag{3.11}$$

η ist die lokale Koordinate des finiten Elementes. Die Ansatzfunktionen sind im lokalen Koordinatensystem definiert, sodass eine Transformation notwendig ist.

$$\mathbf{x}(\eta) = \sum_{i=1}^{m} S^i(\eta_1, \eta_2, \eta_3) x^i \tag{3.12}$$

Bereits mit Verschiebungsansätzen geringer Ordnung können sehr komplexe Formen approximiert werden. Die gewählten Ansätze müssen C_0 konform sein, d.h. dass es nicht zum Klaffen zwischen den Elementübergängen kommt.

Wenn die Ordnung der Ansatzfunktion für die Verschiebung und Geometrie dieselbe ist, handelt es sich um einen isoparametrischen Ansatz.

Für die Lösung der ortsabhängigen Gleichungen werden die Elementsteifigkeitsmatrix $\mathsf{K_e}$ und der Vektor der Knotenkräfte $\mathbf{p_e}$ benötigt. Diese berechnen sich mittels der Ansatzfunktionen wie folgt:

$$\mathsf{K}_e = \int_{\Omega_e} (\mathsf{D}\mathbf{S})^T \mathsf{E} (\mathsf{D}\mathbf{S}) d\Omega \tag{3.13}$$

$$\mathbf{p}_e = \int_{\Omega_e} \mathbf{S}^T \mathbf{p}_{\mathbf{\Omega}} d\Omega + \int_{\Gamma_e} \mathbf{S}^T \mathbf{p} d\Gamma \tag{3.14}$$

Die Integration der Integralgleichung wird numerisch mit Hilfe der Gauß-Quadratur [122] durchgeführt. Die Elementmatrizen werden unter Berücksichtigung der Knotenindices aufaddiert zur Systemgleichung

$$\mathsf{K}\mathbf{u} = \mathbf{p} \tag{3.15}$$

mit den Knotenkräften $\mathbf{p}$ und den Zustandsgrößen $\mathbf{u}$.

3.3 Zeitabhängiges Verhalten nach [98]

Zur Bestimmung der Zustandsgrößen wurde bisher das Gleichungssystem $\mathsf{K}\mathbf{u} = \mathbf{p}$ gelöst. Jetzt wird dieses System mit einem Massen- und Dämpfungsterm erweitert, um dynamische Lasten auf die Struktur wirken lassen zu können. In Kapitel 4 wird die Kopplung von Fluid und Struktur erfolgen. Hier wirken dynamische Lasten aus dem Fluid auf die Struktur ein. Für den nicht statischen Fall lautet das Gleichungssystem:

$$\mathsf{M}\ddot{\mathbf{u}}(t) + \mathsf{D}\dot{\mathbf{u}}(t) + \mathsf{K}\mathbf{u}(t) = \mathbf{p}(t) \tag{3.16}$$

Die Vektoren $\ddot{\mathbf{u}}(t)$ und $\dot{\mathbf{u}}(t)$ enthalten die gleichen Zustandsgrößen wie der Vektor $\mathbf{u}(t)$, diese jedoch nach der Zeit differenziert.

Die Systemmatrix M enthält die konsistente Massenverteilung. Diese wird aus den Elementmassenmatrizen M_e addiert:

$$\mathsf{M}_e = \int_{V_e} \mathbf{S}^T \rho \mathbf{S} dV \tag{3.17}$$

Die Dämpfung wird in den numerischen Simulationen nicht berücksichtigt.

Die Lösung der Differentialgleichung 3.16 folgt hier dem in [98] beschriebenen Ansatz. Es werden auf der Zeitachse die Stützstellen $t_0, t_1, ...$ eingeführt. Diese Stützstellen besitzen den gleichen Abstand h. An jeder Stützstelle werden die Funktionswerte $u_n, \dot{u}_n, \ddot{u}_n$ als Stützwerte eingeführt. Für den Zeitschritt i wird eine normalisierte Variable s eingeführt.

$$t = t_n + sh, \quad h = t_{n+1} - t_n, \quad 0 \leq s \leq 1 \tag{3.18}$$

Für den Verlauf der Variablen $u, \dot{u}, \ddot{u}$ sowie der Einwirkung p werden unabhängige Taylorreihen bis zur Genauigkeit $\dddot{u}$ aufgestellt. Es werden die Newmarkparameter β und γ eingeführt, sodass der Abbruch der Taylorreihe zu möglichst kleinen Fehlern führt. Die Substitution der Taylorreihenapproximation in Gleichung 3.16 zum Zeitpunkt $s = \theta$ führt zu einem Gleichungsystem der Form:

$$\mathsf{A}_\theta \mathbf{u}_\theta = \mathbf{p}_\theta \tag{3.19}$$

mit der Matrix A_θ:

$$\mathsf{A}_\theta = \mathsf{M}\frac{1}{\gamma(\theta h)^2} + \mathsf{D}\frac{\beta}{\gamma(\theta h)} + \mathsf{K} \tag{3.20}$$

und dem Vektor $\mathbf{p}_\theta$

$$\begin{aligned} \mathbf{p}_\theta \quad = \quad & \mathbf{p}_i(1-\theta) + \mathbf{p}_{i+1}\theta + \\ & \mathsf{M}\left(\left(1 - \tfrac{1}{2\gamma}\right)\ddot{\mathbf{u}}_i - \tfrac{1}{\gamma\theta h}\dot{\mathbf{u}}_i - \tfrac{1}{\gamma(\theta h)^2}\mathbf{u}_i\right) + \\ & \mathsf{D}\left(\theta h\left(1 - \tfrac{\beta}{2\gamma}\right)\ddot{\mathbf{u}}_i + \left(1 - \tfrac{\beta}{\gamma}\right)\dot{\mathbf{u}}_i + \tfrac{\beta}{\gamma\theta h}\mathbf{u}_i\right) \end{aligned} \tag{3.21}$$

Über die Konstanten β, γ, θ werden einige Zeitintegrationsverfahren definiert, deren numerische Eigenschaften mit diesen Parametern gezielt gesteuert werden können. Tabelle 3.1 zeigt eine Übersicht verschiedener Verfahren.

Tabelle 3.1: *Parameter für zeitabhängiges Verhalten [98]*

Verfahren	γ	β	θ
Lineare Beschleunigung	$\frac{1}{2}$	$\frac{1}{6}$	1.0
Wilson	$\frac{1}{2}$	$\frac{1}{6}$	≥ 1.37
Newmark	$\geq \frac{1}{2}$	$\leq \frac{1}{4}\left(\gamma + \frac{1}{2}\right)^2$	1.0

Für die gesuchten Größen $u_{n+1}, \dot{u}_{n+1}, \ddot{u}_{n+1}$ ergeben sich folgende Gleichungen:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{1}{\theta^3}(\mathbf{u}_\theta - \mathbf{u}_n) + \left(1 - \frac{1}{\theta^2}\right)h\dot{\mathbf{u}}_n + \frac{1}{2}\left(1 - \frac{1}{\theta}\right)h^2\ddot{\mathbf{u}}_n \tag{3.22}$$

$$\dot{\mathbf{u}}_{n+1} = \frac{\gamma}{\beta\theta^3 h}(\mathbf{u}_\theta - \mathbf{u}_n) + \left(1 - \frac{\gamma}{\beta\theta^2}\right)\dot{\mathbf{u}}_n + \left(1 - \frac{\gamma}{2\beta\theta}\right)h\ddot{\mathbf{u}}_n \tag{3.23}$$

$$\ddot{\mathbf{u}}_{n+1} = \frac{1}{\beta\theta^3 h^2}(\mathbf{u}_\theta - \mathbf{u}_n) - \frac{1}{\beta\theta^2 h}\dot{\mathbf{u}}_n + \left(1 - \frac{1}{2\beta\theta}\right)\ddot{\mathbf{u}}_n \tag{3.24}$$

3.4 Struktursimulation mit Balkenelementen

Für das eindimensionale Balkenelement werden Ansatzfunktionen dritten Grades mit vier Freihheitsgraden (Knotenverdrehung und -verschiebung) verwendet:

$$S = \begin{bmatrix} 1 - 3\eta^2 + 2\eta^3 \\ l_e(-\eta + 2\eta^2 - \eta^3) \\ 3\eta^2 - 2\eta^3 \\ l_e(\eta^2 - \eta^3) \end{bmatrix} \tag{3.25}$$

Die Elementsteifigkeitsmatrix K_e mit der Steifigkeit EI und der Elementlänge l_e ist somit:

$$K_e = \frac{EI}{l_e} \begin{bmatrix} \frac{12}{l_e^2} & -\frac{6}{l_e} & -\frac{12}{l_e^2} & -\frac{6}{l_e} \\ -\frac{6}{l_e} & 4 & \frac{6}{l_e} & 2 \\ -\frac{12}{l_e^2} & \frac{6}{l_e} & \frac{12}{l_e^2} & \frac{6}{l_e} \\ -\frac{6}{l_e} & 2 & \frac{6}{l_e} & 4 \end{bmatrix}. \tag{3.26}$$

und die Elementmassenmatrix M_e mit der längenbezogenen Masse $m = \rho A$):

$$M_e = m l_e \begin{bmatrix} \frac{13}{35} & l_e\frac{11}{210} & \frac{9}{70} & -l_e\frac{13}{420} \\ l_e\frac{11}{210} & l_e^2\frac{1}{105} & l_e\frac{13}{420} & -l_e^2\frac{1}{140} \\ \frac{9}{70} & l_e\frac{13}{420} & \frac{13}{35} & -l_e\frac{11}{210} \\ -l_e\frac{13}{420} & -l_e^2\frac{1}{140} & -l_e\frac{11}{210} & l_e^2\frac{1}{105} \end{bmatrix}. \tag{3.27}$$

3.5 Struktursimulation mit dreidimensionalen Finiten Elementen hoher Ordnung

Der in dieser Arbeit verwendete Strukturlöser ADHOC hat einen strikt dreidimensionalen Ansatz für dünnwandige Strukturen. Hier gibt es grundsätzlich zwei Möglichkeiten. Entweder müssen große Seitenverhältnisse in Kauf genommen werden, oder aber sehr viele isotrope Elemente eingesetzt werden. Standardelemente niedriger Ordnung sind sehr empfindlich gegenüber großen Seitenverhältnissen, womit nur die Möglichkeit der ineffizienten Diskretisierung mit vielen Elementen bleibt.

Im Gegensatz hierzu, bewältigen Elemente höherer Ordnung große Seitenverhältnisse problemlos [122, 123], vorausgesetzt der Polynomgrad ist hinreichend hoch.

In [21, 23, 101] werden Hexaederelemente hoher Ordnung für dünnwandige Strukturen beschrieben. Gekrümmte Strukturen (siehe Abbildung 3.1) können durch passende Mapping Techniken [122, 123] dargestellt werden.

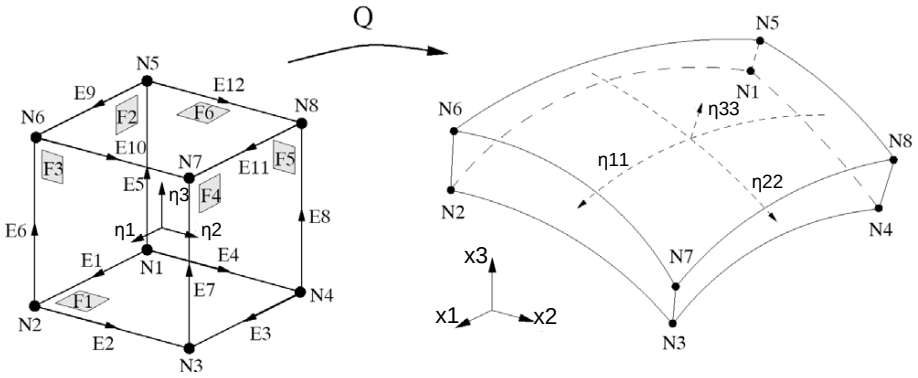


Abbildung 3.1: *Diskretisierung dünnwandiger Strukturen mit Hexaederelementen*

Im Kontext von Elementen höherer Ordnung kann zudem eine effiziente räumliche Diskretisierung durch Einsatz anisotroper Ansatzfunktionen vorgenommen werden. Hier wird der Ansatz aus [21, 123] genutzt, um verschiedene Polynomgrade für die verschiedenen lokalen Richtungen des Hexaederelements zu definieren. Um den Berechnungsaufwand zu senken, werden beispielsweise bei einer schalenartigen Struktur, wie in Abbildung 3.1 dargestellt, höhere Polynomgrade für die Ebenenrichtung und niedrige Polynomgrade für die Dicke verwendet. Weiterhin ist es nicht nur möglich, verschiedene Polynomgrade für die lokalen Richtungen zu definieren. Auch die Polynomgrade für die verschiedenen Komponenten des Verschiebungsfeldes können den Anforderungen entsprechend elementweise gewählt werden. Bei Anwendung dieses Ansatzes geht der Modellfehler der Platten- und Schalentheorie in den Diskretisierungsfehler der dreidimensionalen Approximation über. Hauptvorteil ist, dass dieser Fehler durch verschiedene Polynomgrade über die Dicke kontrolliert werden kann. Bei festen kinematischen Annahmen ist diese Kontrolle nicht möglich. Während für die räumliche Diskretisierung des Strukturproblems Finite Elemente höherer Ordnung zum Einsatz kommen, wird die zeitliche Diskretisierung mit der Methode nach Newmark und der Generalized-α-Methode vorgenommen, welche beide zweiter Ordnung genau sind, letztere jedoch auch bei Einsatz numerischer Dämpfung von hohen Frequenzen [16].

4 Bidirektionale Fluid-Struktur-Interaktion

Bei der Fluid-Struktur-Interaktion liefert die Verformung der Struktur eine veränderte Geometrie für das Fluid. Diese wird durch die in Kapitel 2.4 beschriebenen Algorithmen im Fluidgebiet diskretisiert. Die Ermittlung der Kräfte, die als Randbedingungen für die Strukturberechnung dienen, erfolgt hingegen im Fluidgebiet. Die verschiedenen Varianten, die im Rahmen der Lattice-Boltzmann-Methode existieren, werden im nachfolgenden Abschnitt beschrieben. Bevor die zur Interaktion mit dem Fluid- und Strukturlöser notwendigen Kopplungsalgorithmen vorgestellt werden, zeigt Abschnitt 4.2 die Umrechnung der physikalischen Größen.

4.1 Kraftberechnung

Es gibt zwei Möglichkeiten zur Kraftberechnung: Ermittlung über (a) den Impulsaustausch oder (b) die Integration der Drücke bzw. Spannungen. Die Oberfläche einer Struktur wird dabei durch Polygone in 2-D bzw. Dreiecksnetze in 3-D dargestellt. Eine detaillierte Beschreibung und Vergleich beider Methoden erfolgt in [87].

4.1.1 Impulsaustausch

Die Kraft, die auf ein Objekt wirkt (siehe Abbildung 4.1), wird an jedem Randknoten mit Hilfe des Impulsaustausches zwischen Partikelverteilungen, die auf ein (bewegtes) Objekt und dem Objekt selbst treffen, berechnet [94]:

$$\Delta \mathbf{F}_{\mathbf{i,Knoten}} = \frac{\Delta x^3}{\Delta t}(\mathbf{e}_i - \mathbf{u}_B)(f_i^t + f_I^{t+1}). \tag{4.1}$$

Die Verteilungen $f_I(t+1)$ wird entsprechend Gleichung 2.30 bzw. Gleichung 2.31 bestimmt. $\mathbf{u}_B$ ist hierbei die Geschwindigkeit des bewegten Randes und $\mathbf{e}_I = -\mathbf{e}_i$ sind die LB-spezifischen, mikroskopischen Geschwindigkeiten, die dem jeweiligen diskreten Geschwindigkeitsmodell [100] entsprechen. Abbildung 4.1 zeigt die „Links“ des LB-Gitters, die bei der Kraftermittlung auf die Struktur zu berücksichtigen sind.

Die Kraftanteile $\Delta \mathbf{F}_{\mathbf{i,Knoten}}$, die auf den Rand der Struktur zwischen den Interfacepunkten A und B wirken, werden mittels linearer Interpolation gewichtet und aufsummiert, um die Kräfte an diesen Punkten zu ermitteln. Als Gewicht für die Kraftintegration für Punkt A wird der normalisierte Abstand zwischen dem Schnittpunkt S und dem Strukturpunkt A verwendet. Alle Fluidknoten mit einem „Link“, der den Rand schneidet, tragen zur resultierenden Kraft bei. Diese Methode ist impulserhaltend.

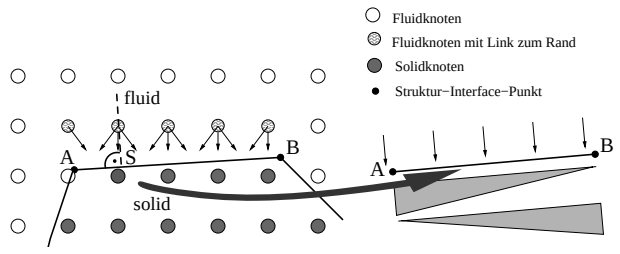


Abbildung 4.1: *Interpolationsschema zur Berechnung der Kräfte*

4.1.2 Spannungsintegration

Die Impulsaustauschmethode funktioniert für die Ermittlung integraler Kräfte auf große Strukturelemente, aber sie ist nicht praktikabel zur Berechnung der Kräfte, wenn die Ausdehnung der Randelemente in der Größenordnung des Fluid-Berechnungsgitters liegt. Inkorrekte lokale Kräfte sind die Folge, wenn zu wenige „Links“ zur Kraftberechnung beitragen. Für diesen Fall ist die Spannungsintegrationsmethode vorzuziehen. Im Rahmen dieser Arbeit wurde das nachfolgende Schema zur Spannungsinterpolation bei der Kraftermittlung entwickelt.

Ein Vorteil gegenüber konventionellen CFD-Lösern ist, dass der Spannungstensor lokal ermittelt werden kann. Der Spannungsdeviator $S_{\alpha\beta}$ mit skalarem Druck wird folgendermaßen berechnet:

$$S_{\alpha\beta} = -c_s^2 \rho \delta_{\alpha\beta} + \sigma_{\alpha\beta} \tag{4.2}$$

Die Spannung $\sigma_{\alpha\beta}$ wird aus den Nichtgleichgewichtsanteilen berechnet. δ ist das Kronecker-Delta und D die Dimension.

$$\sigma_{\alpha\beta} = \left(1 - \frac{\Delta t}{2\tau}\right) \sum_{i=1}^{8} f_i^{neq} \left(e_{i\alpha} e_{i\beta} - \frac{1}{D} \mathbf{e}_i \mathbf{e}_i \delta_{\alpha\beta}\right) \tag{4.3}$$

Die Spannungen an den Randknoten werden über eine Extrapolation berechnet. In Abbildung 4.2 sind die 16 verschiedenen Fälle für den Aufenthalt eines Punktes im Fluidgebiet dargestellt. Die verschiedenen Fälle entsprechen denen des Marching-Squares (2-D) bzw. Marching-Cubes (3-D) Algorithmus [84]. Diese sind bekannt aus der Computergrafik und werden zur Generierung von Isolinien und Isoflächen von kartesischen Gittern verwendet.

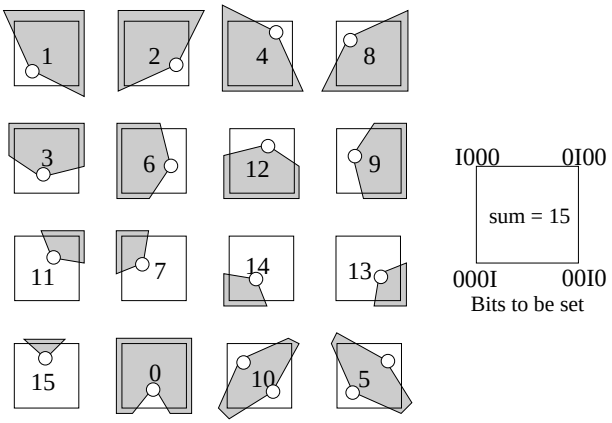


Abbildung 4.2: *Extrapolationsfälle in 2D*

Der jeweilige Index in Abbildung 4.2 repräsentiert die Summe der Bits von Fluidknoten. Da die Ausdehnung der Struktur größer ist als der Gitterabstand, können die Fälle 10 und 5 ausgeschlossen werden. Für den Fall 15 ist die Spannungsinterpolation bilinear, da hier alle zur Interpolation beitragenden Knoten im Fluidzustand sind. Für den selten vorkommenden Fall 0 werden die Kräfte aus dem vorangehenden Zeitschritt verwendet. In den anderen Fällen werden die Spannungen aus dem Fluidgebiet auf die imaginären Knoten im Strukturgebiet linear extrapoliert bevor sie, wie in Abbildung 4.3 dargestellt, bilinear interpoliert werden.

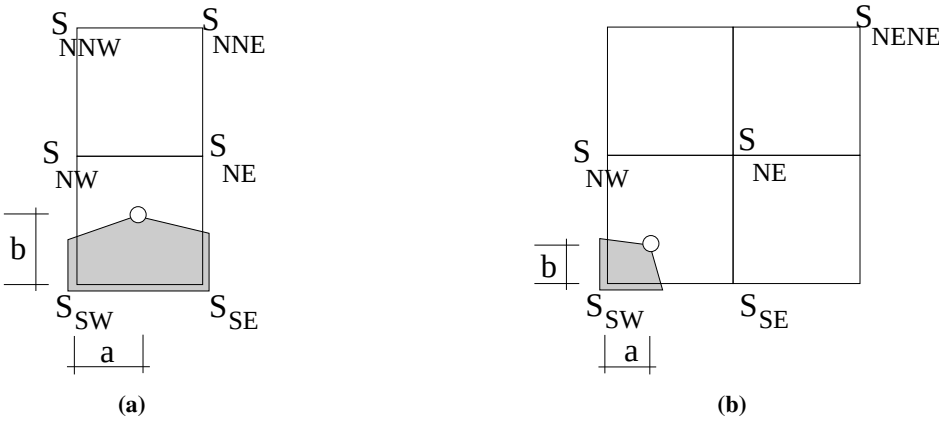


Abbildung 4.3: *Spezifische Extrapolationsfälle*

Für die Fälle aus Abbildung 4.3a und Abbildung 4.3b ergeben sich beispielhaft folgende Extrapolationen:

Fall (a)

$$\begin{aligned} \mathsf{S}_{SW} &= 2 \cdot \mathsf{S}_{NW} - \mathsf{S}_{NNW} \\ \mathsf{S}_{SE} &= 2 \cdot \mathsf{S}_{NE} - \mathsf{S}_{NNE} \end{aligned} \tag{4.4}$$

Fall (b)

$$\mathsf{S}_{SW} = 2 \cdot \mathsf{S}_{NE} - \mathsf{S}_{NENE} \tag{4.5}$$

Nach der Extrapolation werden die Spannungen S mit den normalisierten Wichtungsfaktoren a und b ($0 \leq \{a, b\} \leq 1$) (siehe Abbildung 4.3) bilinear interpoliert:

$$\mathsf{S}_{Punkt} = (1-a)\,(1-b)\,\mathsf{S}_{SW} \;+\; a\,(1-b)\,\mathsf{S}_{SE} \;+\; a\,b\,\mathsf{S}_{NE} \;+\; (1-a)\,b\,\mathsf{S}_{NW} \tag{4.6}$$

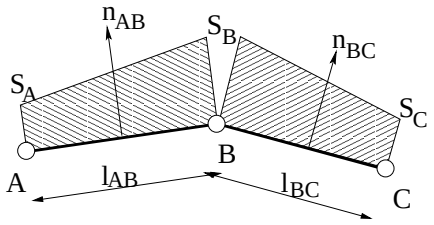


Abbildung 4.4: *Kraftberechnung durch Spannungsintegration*

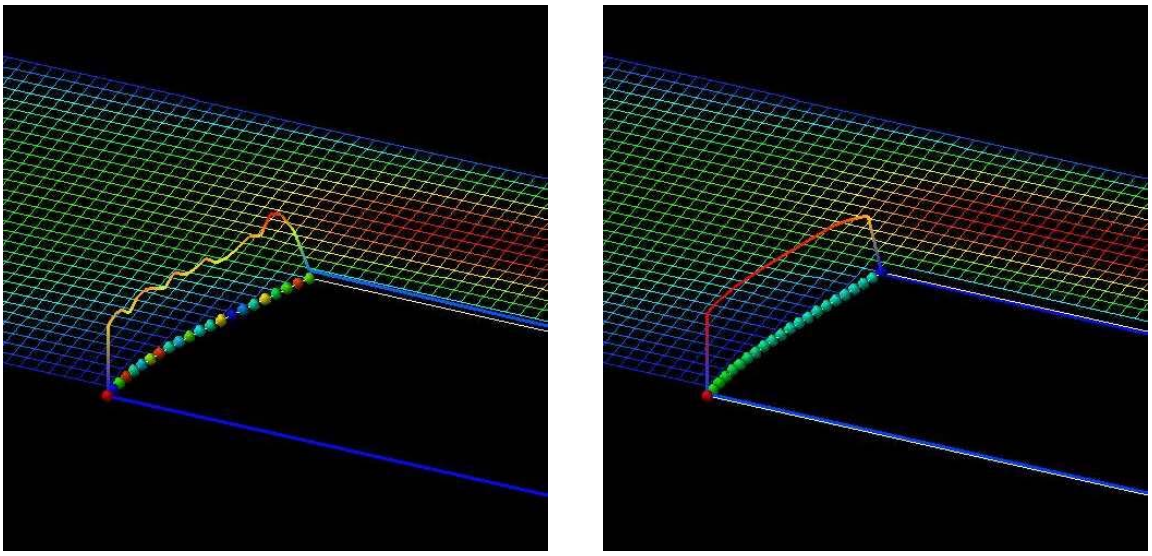
Durch Multiplikation des Spannungstensors mit der Normalen des Randes Γ und der Integration entlang des Randes erhält man den Lastvektor $\mathbf{F}$.

$$\mathbf{F} = \int_{\Gamma} \mathsf{S} \cdot \mathbf{n}_{Rand}\, d\Gamma. \tag{4.7}$$

Des Weiteren wird angenommen, dass der Spannungsverlauf zwischen zwei Randknoten linear ist (siehe Abbildung 4.4). Der Lastvektor $\mathbf{F}_B$ am Punkt B wird berechnet, indem Gleichung 4.7 von Punkt $\frac{1}{2}(B+C)$ bis Punkt $\frac{1}{2}(A+B)$ integriert wird:

$$\mathbf{F}_B = \left(\frac{1}{4}\mathsf{S}_A + \frac{3}{4}\mathsf{S}_B\right)\mathbf{n}_{AB}\frac{1}{2}l_{AB} + \left(\frac{3}{4}\mathsf{S}_B + \frac{1}{4}\mathsf{S}_C\right)\mathbf{n}_{BC}\frac{1}{2}l_{BC} \tag{4.8}$$

In Abbildung 4.5 ist der qualitative Unterschied beider Methoden anhand eines angeströmten Kragarms dargestellt. Der Spannungsverlauf ist bei der Spannungsintegration formerhaltend, jedoch nicht impulserhaltend, da die Einzelimpulse nicht mehr berücksichtigt werden.



(a) Impulsaustausch **(b)** Spannungsintegration

Abbildung 4.5: *Kraftberechnung*

4.2 Transformation physikalischer Größen

Für Fluid-Struktur-Interaktionsprobleme sind folgende dimensionslose Kennzahlen notwendig:

- die Aerolastische Zahl $Ae = \frac{E_s}{\rho_f u_f^2}$
- die Reynoldszahl $Re = \frac{u_f H}{\nu}$
- das Dichteverhältmis $\beta = \frac{\rho_s}{\rho_f}$
- die Querdehnzahl (Poissonzahl) der elastischen Struktur ν_s

E_s ist das Elastizitätsmodul. ρ_s und ρ_f sind die Dichte der Struktur und des Fluids. u_f ist eine Referenzgeschwindigkeit, H eine Referenzlänge und ν ist die kinematische Viskosität.

Bei der Lattice-Boltzmann-Methode werden alle Größen in sogenannten Lattice-Einheiten verwendet. Das heisst, das System ist derart skaliert, dass $\Delta t = \Delta x = 1$ gilt. Aus diesem Grund müssen alle Größen zwischen den zwei verschiedenen Systemen, bezeichnet als *LB* und *real*, skaliert werden. Die Skalierung der Kräfte vom Fluidlösersystem (LB) zum realen System wird über das Gleichsetzen des dimensionslosen Kraftbeiwertes c_d (Dragkoeffizient),

$$c_d = \frac{2F}{\rho u^2 D} \tag{4.9}$$

der in beiden Systemen identisch sein muss, bewirkt. Das Ergebnis ist:

$$\mathbf{F}_{real} = \mathbf{F}_{LB} \cdot \frac{H_{real} \cdot \rho_{real} \cdot u_{real}^2}{H_{LB} \cdot \rho_{LB} \cdot u_{LB}^2} \tag{4.10}$$

mit der Referenzhöhe H, der Referenzdichte ρ und der Referenzgeschwindigkeit u.

Ist keine Referenzgeschwindigkeit gegeben, wie beispielsweise bei einem ruhenden Fluid, kann man die Kräfte mit folgender Formel, bei der der Kraftbeiwert mit dem Quadrat der Reynoldszahl multipliziert wird, berechnen:

$$\mathbf{F}_{real} = \mathbf{F}_{LB} \cdot \frac{\rho_{real} \cdot \nu_{real}^2 \cdot H_{LB}}{\rho_{LB} \cdot \nu_{LB}^2 \cdot H_{real}} \tag{4.11}$$

Die Transformation der Kräfte gilt für zweidimensionale Berechnungen. Für dreidimensionale Ermittlungen kommt der Faktor $\frac{H_{real}}{H_{LB}}$ hinzu. Die Transformationsformel lautet:

$$\mathbf{F}_{real} = \mathbf{F}_{LB} \cdot \frac{H_{real}^2 \cdot \rho_{real} \cdot u_{real}^2}{H_{LB}^2 \cdot \rho_{LB} \cdot u_{LB}^2} \tag{4.12}$$

bzw. für den Fall, dass keine Referenzgeschwindigkeit vorgegeben ist:

$$\mathbf{F}_{real} = \mathbf{F}_{LB} \cdot \frac{\rho_{real} \cdot \nu_{real}^2}{\rho_{LB} \cdot \nu_{LB}^2} \tag{4.13}$$

Für die Zeitskalierung wird die Reynoldszahl in beiden Systemen gleichgesetzt und mit $u = \frac{H}{\Delta t}$ erhält man:

$$\Delta t_{real} = \Delta t T_{LB} \cdot \frac{\nu_{LB} \cdot H_{real}^2}{\nu_{real} \cdot H_{LB}^2} \tag{4.14}$$

Nach der Berechnung der Verschiebungen und der Geschwindigkeiten der Struktur wird die Wandgeschwindigkeit (Randbedingung) für das LB-System folgendermaßen ermittelt:

$$\mathbf{u}_{LB} = \mathbf{u}_{real} \cdot \frac{\nu_{LB} \cdot H_{real}}{\nu_{real} \cdot H_{LB}} \tag{4.15}$$

Weitere dimensionslose Kennzahlen für die Umrechnung vom realen System zum LB-System sind folgende:

- der Druckbeiwert $\alpha = \frac{p}{\rho \cdot u^2}$,
- der Druckbeiwert $\beta = \frac{p}{\rho \cdot g \cdot H}$,
- der Gewichtskraftbeiwert $\gamma = \frac{g \cdot H^3}{\nu^2}$,
- der Gewichtskraftbeiwert $\delta = \frac{g \cdot H}{u^2}$,

Durch Gleichsetzen des Druckbeiwertes im realen und LB-System folgt für den realen Druck:

$$p_{real} = p_{LB} \cdot \frac{\rho_{real} \cdot u_{real}^2}{\rho_{LB} \cdot u_{LB}^2} \tag{4.16}$$

Folgende Formel ergibt sich zur Umrechnung der Gravitationskraft zwischen den beiden Systemen:

$$g_{LB} = g_{real} \cdot \frac{u_{LB}^2 \cdot H_{real}}{u_{real}^2 \cdot H_{LB}} \tag{4.17}$$

bzw. für den Fall, dass keine Referenzgeschwindigkeit vorgegeben ist:

$$g_{LB} = g_{real} \cdot \frac{\nu_{LB}^2 \cdot H_{real}^3}{\nu_{real}^2 \cdot H_{LB}^3} \tag{4.18}$$

4.3 Kopplungsansatz

Der Rand der Struktur wird durch Polynome höherer Ordnung beschrieben, die innerhalb der Elemente C^p- und über deren Grenzen hinweg C^0-stetig sind. Der Rand im Fluid hingegen wird auf einem hierarchisch strukturierten Gitter beschrieben. Diese unterschiedlichen Diskretisierungen werden mit einem Interfacenetz vereint. Grundidee ist hierbei, ein Interfacennetz zu definieren, auf dem die Lasten einerseits für den Fluidlöser einfach zu ermitteln sind und andererseits im Strukturlöser integriert werden können. Die Verschiebungen bzw. Geschwindigkeiten sollten nach Möglichkeit auf demselben Netz ausgetauscht werden.

Als Interfacenetz wurde deshalb ein bewegtes Oberflächennetz bestehend aus ebenen Dreiecken gewählt. Jedem Knoten des Interfacenetzes werden die Werte für die Geschwindigkeit, der Lastvektor und andere physikalische Größen, die für den Austausch benötigt werden, zugeordnet. Das Interfacenetz kann auf zwei Arten definiert werden. (A) Entweder werden die Knoten an den Gaußschen Integrationspunkten des p-FEM-Lösers erzeugt oder (B) es wird ein äquidistantes Netz über die Oberfläche gelegt. Die Knoten des Interfacenetzes werden trianguliert.

Ein Beispiel für ein Interfacenetz und dessen zugehörige Kopplung ist in der Abbildung 4.6 dargestellt. Für den zweidimensionalen Fall reduziert sich das Interfacenetz zu einem Polygon.

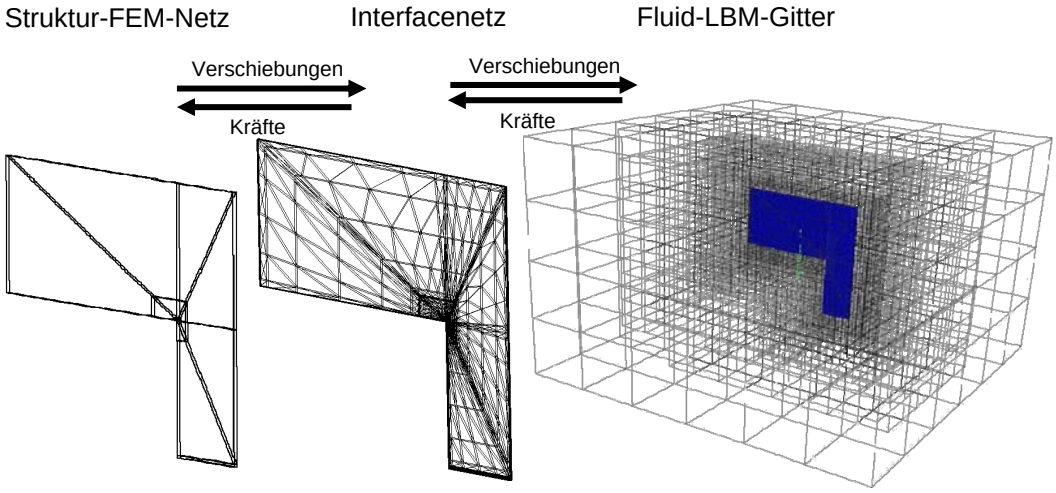


Abbildung 4.6: *Kopplung mittels Interfacenetz an Gaußschen Integrationspunkten*

Die Kopplung wurde mit einer eigens hierfür entwickelten Bibliothek basierend auf einem Client-Server Konzept verwirklicht [8]. Ein Masterprozess verwaltet dabei den Austausch der Daten zwischen dem Fluid- und Strukturcode. Die Kräfte und Verschiebungen werden an den Eckknoten eines triangulierten Netzes ausgetauscht (vgl. Abbildung 4.6).

Vor diesem Hintergrund bestimmen drei wichtige Faktoren die Konsistenz beim Datentransfer:

(1) Typ des Interfacenetzes (äquidistantes Netz oder Gaußpunktnetz)

(2) Art der Kraftextrapolation auf die Strukturoberfläche (siehe Kapitel 4.1.2)

(3) Integration der Kräfte für den Kraftvektor der Strukturelemente (Berücksichtigung der verformten Struktur und Aufbau des Lastintegrals)

Betrachten wir zunächst den Fall (A) in welchem das Oberflächennetz durch die Gaußpunkte des p-FEM Lösers definiert ist. Die lokale Größe der Dreiecke hängt direkt von der Verteilung der Gausspunkte ab. Diese nimmt zu den Elementrändern hin überproportional ab. Es ergeben sich somit sehr kleine Dreiecke in der Nachbarschaft der p-Elementgrenzen und überproportional große Abstände in der Mitte der p-Elemente. Ideal im Hinblick auf den zu erwartenden Fehler im Last- und Verschiebungstransfer aber wäre es, wenn die Verteilung der Eckpunkte identisch mit oder mindestens in derselben Größenordnung wie die Fluiddiskretisierung läge.
Aus diesem Grund wurde ein Kopplungsnetz eingeführt, welches unabhängig von den Gaußpunkten ist. Es ist äquidistant und die Größe der Dreiecke liegt in derselben Größenordnung wie der Abstand der Gitterpunkte in der Fluiddiskretisierung.

Da die Randgeometrie natürlicherweise von der Struktur vorgegeben ist, wird auch das Interfacenetz vom Strukturlöser erstellt und dem Fluidlöser vorgegeben. Auf dessen Knoten werden nun die Spannungsvektoren gemäß der in Kapitel 4.1 beschriebenen Vorgehensweise ermittelt und an den Strukturlöser übertragen. Dieser integriert dann die Lasten in der verformten Konfiguration, die durch den vorherigen Zeitschritt (oder einen Prediktor) vorgegeben ist.

Die neuen Verschiebungen werden direkt an den Knoten des Interfacenetzes vom Strukturlöser ermittelt und zum Fluid übertragen.

Die Knoten des Interfacenetzes liegen exakt auf dem Rand der Struktur und im Fluidlöser erscheint diese Verschiebung zwischen den Knoten linear. Der Rand der Struktur hingegen wird durch stückweise Polynome höherer Ordnung beschrieben. Diese Ungleichheit der Randbeschreibung kann unter Umständen kinetische Energie am Interface einführen. Zur Minimierung dieses Fehlers ist eine globale L_2-artige Projektion vorstellbar. Dies könnte im Hinblick auf den Energieerhalt am Interface zu weiteren Verbesserungen führen, wurde in dieser Arbeit aber als untergeordnet betrachtet und nicht realisiert.

Zunächst wird der explizite Kopplungsansatz beschrieben, mit dem auch geometrisch nichtlineares Strukturverhalten abgebildet werden kann.

4.3.1 Expliziter Kopplungsalgorithmus

Die explizite Kopplung mit Subcycling ist in Abbildung 4.7 dargestellt und kann folgendermaßen zusammengefaßt werden:

Algorithmus 4 Expliziter Kopplungsalgorithmus

(1) Der Fluidlöser berechnet den Lastvektor für die Knoten des Interfacenetzes

(2) Die Kräfte werden mit Hilfe der Kopplungsbibliothek zum Strukturlöser gesendet

(3) Die Struktur integriert die Lasten

(4) Der Strukturlöser berechnet die Verschiebungen, die an den Fluidlöser mittels Kopplungsbibliothek geschickt werden

(5) Im Fluidlöser erfolgt eine Zeitinterpolation der Position der Interfacenetzgeometrie sowie eine dynamische Anpassung der Gittertopologie

(6) Schritt 5 wird wiederholt für alle internen Iterationen des gestaffeltes Zeitschrittschema

(7) Schritt 5 und 6 werden für die Anzahl $n-1$ der Fluid Subcyclingschritte wiederholt

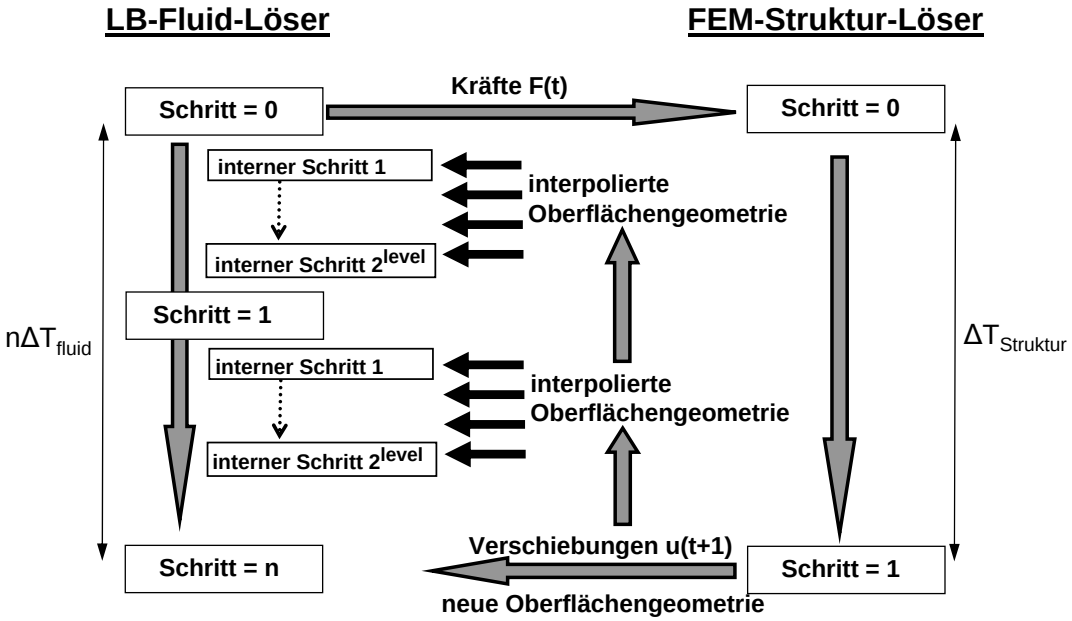


Abbildung 4.7: *Expliziter Kopplungsalgorithmus*

Bevor die Details des impliziten Schemas diskutiert werden, wird der explizite Kopplungsansatz nochmal in komprimierter Form in Abbildung 4.8 gezeigt. Zum Zeitschritt t berechnet der Fluidlöser den Lastvektor an den Knoten des Interfacenetzes. Infolge der Lasten werden die Verschiebungen vom Strukturlöser

berechnet. Gewöhnlich ist der Zeitschritt des Fluidlösers kleiner, als der des Strukturlösers. Internes Subcycling wird zur Integration des Fluidproblems von Zeitschritt t bis $t + 1$ angewandt, wobei die geometrische Position mittels linearer Interpolation ermittelt wird.

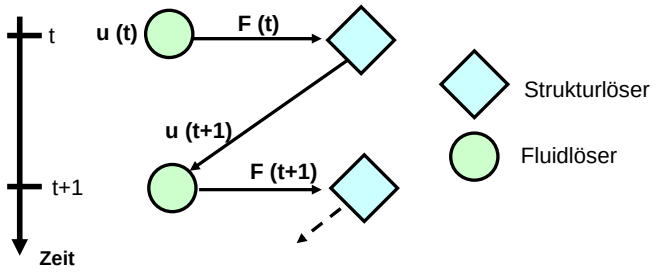


Abbildung 4.8: *Expliziter Kopplungsalgorithmus*

4.3.2 Impliziter Kopplungsalgorithmus

Im Gegensatz zum expliziten Schema, wird beim impliziten Schema über denselben Zeitschritt iteriert bis entweder die Kräfte und/oder die Verschiebungen der letzten zwei Wiederholungen konvergieren. Diese Art der impliziten Iteration ist zur Richardson Iteration [90] hinsichtlich der Freiheitsgrade am Interface ähnlich. Bei schlecht konditionierten Konfigurationen kann eine Unterrelaxation eingesetzt werden, um eine Konvergenz zu ermöglichen. Dies führt aber immer zu einer Verlangsamung der Konvergenz. Bei gut konditionierten Systemen kann mittels Überrelaxation eine bessere Konvergenz erreicht werden. Die Relaxation kann auch adaptiv pro Zeitschritt vorgenommen werden, wobei die gesamte Palette der Vektoriterationsbeschleuniger eingesetzt werden kann. In der FSI hat sich hierbei insbesondere der Beschleuniger nach Aitken für effizient erwiesen, die in den Arbeiten von Mok [90] und Küttler [68, 69] eingehend untersucht und gradientenbasierten Verfahren gegenübergestellt wurde.

Im klassischen Ansatz der Diskretisierung der inkompressiblen Navier-Stokes-Gleichungen (FEM, FVM) muss innerhalb jeden Iterationsschrittes gewöhnlich das gesamte Berechnungsgebiet neu berechnet werden. Als Ergebnis ist jeder Iterationsschritt der impliziten Kopplung genauso rechenintensiv wie ein expliziter Schritt.

Infolge der expliziten und der schwach kompressiblen Natur der LB-Methode ist es möglich, die Subiteration des impliziten Schemas nur auf einer kleinen Region des Ausgangsgitters um das Interfacenetz herum auszuführen (Abbildung 4.9). Zudem zahlt sich der Vorteil der Verwendung eines kartesischen, hierarchisch auf Baumstrukturen organisierten Gitters zur Diskretisierung des Fluidgebietes aus. Ein herausschneiden dieses Subgitters ist hiermit algorithmisch sehr einfach.

Innerhalb der Subiterationen wird somit nur die betreffende Subregion zum nächsten Zeitschritt berechnet, bis eine Konvergenz der Verschiebungen und/oder Kräfte erreicht wird. Die resultierende Konfiguration wird dann auf das Ausgangsgitter angewendet. Erst danach wird das gekoppelte System zum nächsten Zeitschritt integriert. Die Subiterationen in dieser impliziten Prozedur sind somit weniger rechenintensiv als bei einem impliziten Standardalgorithmus.

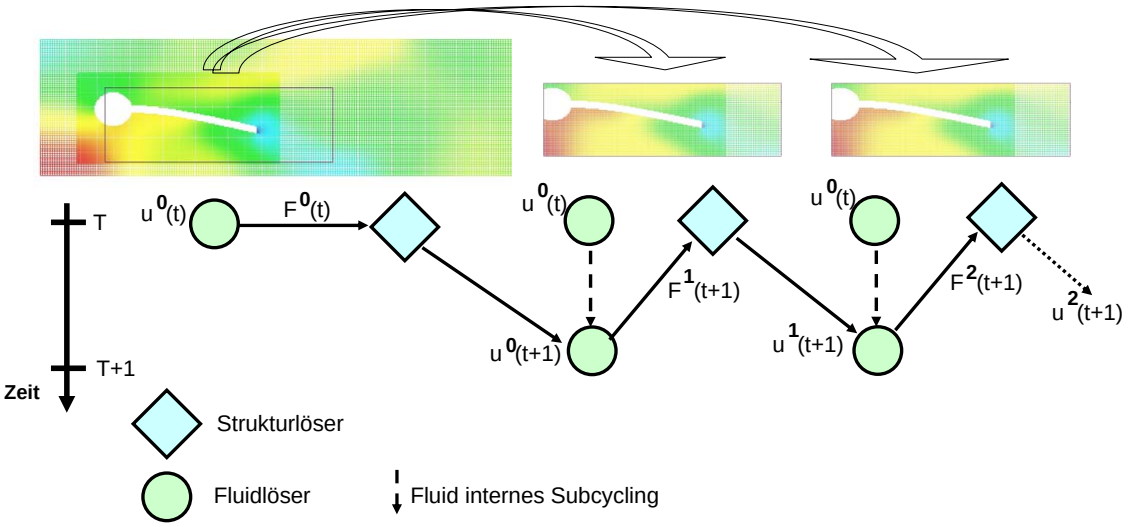


Abbildung 4.9: *Impliziter Kopplungsalgorithmus*

5 Zweidimensionale Testfälle

In diesem Kapitel wird das sukzessive Vorgehen zur Validierung sowohl des Fluidlösers als auch der gekoppelten Fluid-Struktur-Simulation beschrieben. Standardtestfälle, wie die Taylor-Couette-Strömung, die Poiseuille-Strömung oder der DFG Testfall der Zylinderumströmung bei $Re = 20$ und $Re = 100$ [103] wurden von Crouse [17] ausführlich beschrieben und sollen hier nicht weiter aufgeführt werden. In dieser Arbeit wird zunächst ein Benchmark für poröse Medien für einen stationären und einen transienten Fall zum Vergleich mit anderen numerischen Verfahren durchgeführt. Der erste Schritt in Richtung Fluid-Struktur-Wechselwirkung ist die gekoppelte Simulation der Starrkörperbewegung eines oszillierenden Zylinders [88].

Der Lattice-Boltzmann-Strömungslöser VIRTUALFLUIDS, basierend auf unstrukturierten quadtree/octree-artigen Eulergittern, wird gekoppelt mit dem Finite Elemente höherer Ordnung (p-FEM) Strukturmechaniklöser ADHOC, basierend auf einer Lagrangebeschreibung, um bidirektionale Fluid-Struktur-Interaktion (FSI) zu berechnen.

Die Kopplung wurde anhand eines numerischen und eines experimentellen Benchmarks [42, 51], die in der DFG-Forschergruppe 493 *Fluid-Struktur-Wechselwirkung: Modellierung, Simulation, Optimierung* definiert wurden, untersucht.

Die Benchmarkergebnisse wurden in den Veröffentlichungen [37, 38, 65] umfassend dargelegt. Nachfolgend sollen diese nochmals beschrieben und die wichtigsten Erkenntnisse erläutert werden.

5.1 Benchmark für poröse Medien

Einen Beitrag zur Diskussion, ob Lattice-Boltzmann-Methoden als effizienter CFD-Löser betrachtet werden können, liefert der poröse Medien Benchmark. Es wurde hierbei die Genauigkeit und Berechnungseffizienz von zwei Forschungssimulationscodes basierend auf der LB und der Finite Element Methode (FEM) für inkompressible laminare zweidimensionale Strömungsprobleme in komplexen Geometrien verglichen.

Die FEATFLOW Software [26] ist ein (in-)stationärer CFD-Löser, der Multigridalgorithmen nutzt. Der kommerzielle Code CFX [14] ist ein Finite Volumen basiertes CFD Paket, das in Raum und Zeit zweite Ordnung Genauigkeit bietet und dem ein algebraischer Multigrid-Löser zugrunde liegt. Der LB-Löser ist der am Institut für rechnergestützte Modellierung im Bauingenieurwesen entwickelte Forschungsprototyp VIRTUALFLUIDS [37, 129].

LB-Methoden werden genutzt um eine Vielfalt komplexer Strömungen zu simulieren. Aber es gibt nur unzureichende Nachweise über dir Berechnungseffizienz eines LB-CFD-Lösers verglichen mit einem *state-of-the-art*-Löser basierend auf der direkten Diskretisierung der inkompressiblen Navier-Stokes-Gleichungen. Es gibt viele Studien, die die Ergebnisse von Simulationen mit kinetischen und makroskopischen Methoden verglichen haben [82, 103, 95, 4, 61, 72]. Eindeutige Schlussfolgerungen zur Effizienz konnten jedoch nicht gemacht werden, da letzte Modellerweiterungen, wie z. B. lokale Gitterverfeinerung, verbesserte Randbedingungen mit Genauigkeit zweiter Ordnung und Multiple Relaxation Time Modelle (siehe Kapitel 2.1.2) nicht in den kinetischen Simulationsprototyp integriert wurden. Teilweise sind auch die Details des verwendeten LB-Modells nicht bekannt, wie für den kommerziellen Löser POWERFLOW [24].

Zur Vereinfachung der Validierung beschränkt sich der Benchmark auf ein laminares Strömungsregime.

Die Mehrheit der LB-Implementierungen verwendet Matrizenstrukturen, um die Knotenverteilungen zu speichern. Dies erlaubt eine einfache und effiziente Implementierung. Wenn sich viele Hindernisstruk-

turen im Berechnungsgebiet befinden, sind Matrizenstrukturen ineffizient im Hinblick auf den Speicherplatzbedarf. Hier wird eine indirekte Adressierung durchgeführt, bei der Gitterknoten nur dort gesetzt werden, wo es notwendig ist. Um eine ähnliche Effizienz, gemessen in nodal updates per second (NUPS), wie mit dem Matrixcode zu erhalten [111], ist zusätzlicher Programmieraufwand nötig. Entsprechend wurden geeignete Datenstrukturen für topologisch unstrukturierte Gitter entwickelt [31, 111]. Die optimale Datenstruktur hängt vom spezifischen Strömungsproblem mit den spezifischen Randbedingungen ab. Hier werden hierarchische Gitter (Quadtrees in 2-D und Octrees in 3-D), bestehend aus quadratischen Elementen, die rekursiv unterteilt werden, genutzt um eine lokale Gitterverfeinerung zu erhalten. Dieser Ansatz erlaubt eine exponentielle Verfeinerung im Raum, wobei der Berechnungsaufwand bergrenzt bleibt. Diese Datenstrukturen sind auch gut geeignet für Gittervergröberung und erlauben Berechnungen auf adaptiven Gittern [17, 18]. Der C++ Forschungsprototyp *VirtualFluids* liefert $\simeq 4 \times 10^6$ NUPS auf einem 2.0 GHz AMD Opteron 64-Bit Prozessor für das D2Q9-Modell.

Nachfolgend werden einige Eigenschaften des Finiten-Element-Prototypen skizziert. Danach wird das Benchmarkproblem definiert und die Ergebnisse der Berechnung zusammen mit den Netzen präsentiert.

5.1.1 Direkte Diskretizierung der Navier-Stokes-Gleichungen mit Finiten Elementen und Finiten Volumen

FEATFLOW, `www.featflow.de`, ist ein (paralleler) 2-D und 3-D FEM-Code für die Lösung der inkompressiblen Navier-Stokes-Gleichungen

$$\mathbf{u}_t - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \quad \nabla\cdot\mathbf{u} = 0, \tag{5.1}$$

die seperat in Raum und Zeit diskretisiert sind. Die Behandlung von zeitabhängigen Strömungskonfigurationen mit dem PP2D-Modul wird in [134] beschrieben. Die Navier-Stokes-Gleichungen werden durch Standardmethoden zweiter Ordnung, bekannt von der Behandlung gewöhnlicher Differentialgleichungen (Fractional-Step-θ-Schema, Crank-Nicolson-Schema)[134], in der Zeit diskretisiert. Der Raum wird durch die Anwendung spezieller Finiter-Element-Ansätze, die $\tilde{Q}_1/Q_0$ Räume (nichtparametrische Version, siehe [135]) diskretisiert.

Der konvektive Term wird durch ein Upwind-Verfahren (gewichtetes Samarskij-Upwind) stabilisiert. Eine adaptive Zeitschrittsteuerung für diesen impliziten Ansatz wird durch die Bestimmung des lokalen Abbruchfehlers realisiert. Folglich werden Lösungen von verschiedenen Zeitschritten verglichen und das gekoppelte Problem ist innerhalb jeden Zeitschritts gesplittet in skalare Teilprobleme mit dem diskreten Projektionsverfahren [135, 133].

Das produziert eindeutig Probleme in **u** (Burgers-Gleichung) sowie in p (Druck-Poisson). Das nichtlineare Problem in **u** wird mit einer Fixpunktdefektkorrekturmethode und das linearisierte asymmetrische Teilproblem wird durch einen Multigrid-Ansatz gelöst. Ein spezieller Multigrid-Löser mit Jacobi/SOR/ILU-Glättern wird auf das gut konditionierte, lineare Problem in p angewandt. Ein direkter stationärer Ansatz (Modul CC2D, siehe [135]) wird für niedrige Reynoldszahlen genutzt, wie z. B. stationäre Strömungskonfigurationen, die das nichtlineare System nach der FEM-Diskretisierung im Raum (mit Q_2/P_1 Elementen und Stromlinien-Diffusions-Stabilisierung) mit einem Quasi-Newton Löser behandelt. Das resultierende Oseen Teilproblem wird mit direkten Multigrid Ansätzen, basierend auf einem lokalen Pressure Schur Complement Löser („Vanka-artig"), gelöst.

Numerische Details zur Featflow Methodologie und der Softwareimplementierung können auf der Featflow-Webseite und den dortigen Referenzen eingesehen werden.

ANSYS CFX [14] ist ein integriertes Softwaresystem, das diverse und komplexe dreidimensionale Strömungsprobleme löst. Der Multigrid-basierte Fluidlöser liefert Lösungen für inkompressible oder kompressible, stationäre oder transiente, laminare oder turbulente, Ein- und Mehrphasen und Temperaturströmungen in komplexen Geometrien. Die Software benutzt unstrukturierte und blockstrukturierte nicht orthogonale Gitter zur Bereichsdiskretisierung.

5.1.2 Benchmarkdefinition

Die zu betrachtene Strömungsgeometrie ist in Abbildung 5.1 dargestellt. Ein Kanal mit der Höhe H und einer Länge von $4H$ ist mit zylindrischen Objekten mit dem Durchmesser $H/8$ und einem Abstand von $H/8$ besetzt. Für den linken, vertikalen Einflussrand (x=0,y) sowie die oberen (x,y=H) und unteren Wände (x,y=0) werden Dirichlet-Randbedingungen mit einer Geschwindigkeit von $U_0 = (u_x, 0)$ verwendet. Der rechte vertikale Ausflussrand ($x = 4H, y$) hat einen Referenzdruck P_0.

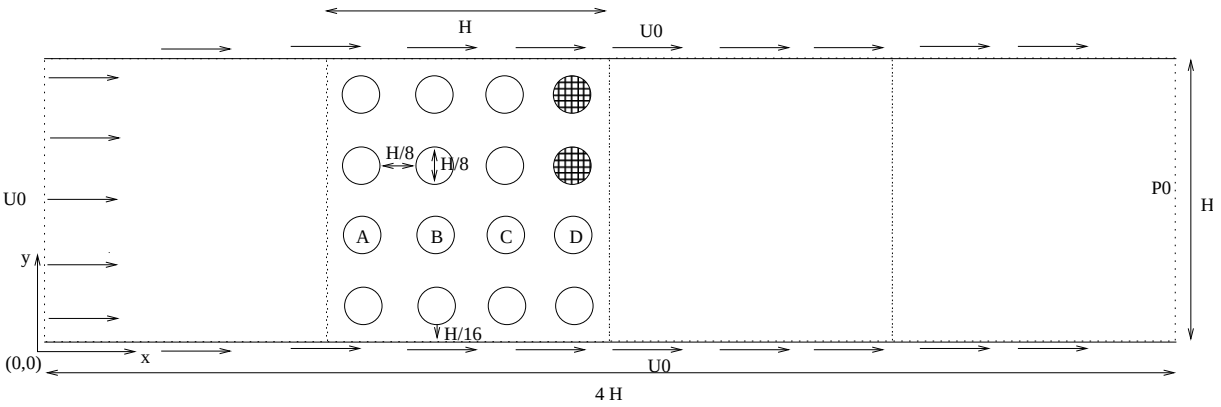


Abbildung 5.1: *Definition des Bechmark-Setup*

Definiert wird das Strömungsproblem über die Ergun'schen Reynoldszahl

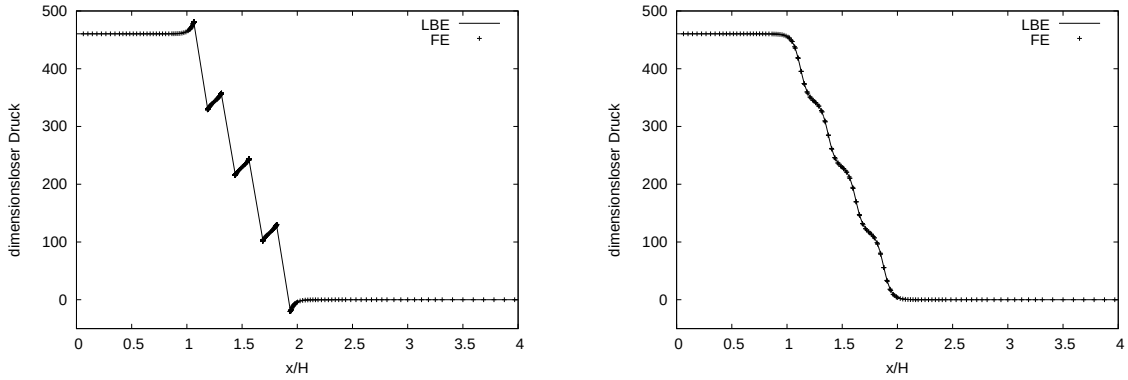
$$Re_E = \frac{U_0 \ D_P}{\nu(1-\epsilon)} \tag{5.2}$$

wo D_P der Zylinderdurchmesser ($H/8$), ν die kinematische Viskosität und ϵ die Porosität ist. Die Porosität ϵ wird für alle Fälle mit 0.75 definiert.

Es werden zwei Testfälle simuliert: $Re_E = 1$ und $Re_E = 200$. Für den $Re_E = 200$ Fall werden die beiden Zylinder rechts oben (kariert in Abbildung 5.1) weggelassen, um die Symmetrie der Strömung zu brechen.

5.1.3 Ergebnisse

Der erste Übereinstimmungstest ist der Vergleich des Druckabfalles durch das System für den stationären Fall $Re_E = 1$ mit zwei verschiedenen Ansätzen. Es wird eine annähernd vergleichbare Anzahl an Freiheitsgraden (DOF=degree of freedom) (LB 549.936 DOF, FEM 297.858 DOF.) gewählt. Abbildung 5.2 zeigt zwei Druckplots entlang zwei verschiedener horizontaler Linien ($y = 0.5/0.625H$). Die Ergebnisse sind praktisch nicht unterscheidbar.



(a) $y = 0.625H$ **(b)** $y = 0.5H$

Abbildung 5.2: *Drucklinien für* $Re_E = 1$ *entlang horizontaler Linien*

Da die Druckbeiwerte (Drag und Lift) sensiblere Größen sind, wurden die entsprechenden Ergebnisse für verschiedene Netze im stationären Zustand und zeitabhängige Strömungen verglichen. Alle CPU-Zeiten wurden auf einem AMD Opteron 64-Bit 2.0 GHz System mit dem Betriebssystem Linux gemessen. Zwei typische Netze sind in Abbildung 5.3 und 5.4 dargestellt. Die Genauigkeit der Ergebnisse, die mit diesen Netzen erzielt wurden, ist vergleichbar (siehe Tabelle 5.1 und 5.2). Die Verfeinerungsbezeichnung für das FE-Netz ist folgendermaßen definiert: der $1 + 2$ Fall (siehe Abbildung 5.4) bezeichnet, dass alle Ausgangselemente mit dem Basislevel 1 gegeben sind und dass einige Basiselemente um ausgewählte Zylinder um zwei weitere Level durch rekursive Partitionierung verfeinert sind. Im LB-Kontext ist ein Gitterlevel l gleichzusetzen mit $\Delta x = H2^{-l}$. Die Ergebnisse der Gitterkonvergenzstudien sind in Tabelle 5.3 und 5.4 sowie in Abbildung 5.5 zusammengefaßt. Die Berechnungszeit steigt quadratisch mit der Anzahl der Freiheitsgrade für den LB-Löser an. Hingegen ist der FEM-Löser in der Asymptotik effizienter, da er linear mit der Anzahl der Freiheitsgrade skaliert (Multigrid-Komplexität).

Tabelle 5.1: *Dragbeiwert: Zylinder A, Referenz 465.58*

Schema	Gitterbezeichnung	#DOF	rel. Fehler[%]	CPU-Zeit[s]
LB	4-6	48.096	1.03	10
LB	4-7	141.696	0.43	93
FEM	0+2	11.774	0.77	6
FEM	1+2	26.922	0.58	53
FEM	2+0	30.642	1.34	33
FEM	2+1	43.314	0.37	38

Tabelle 5.2: *Liftbeiwert: Zylinder A, Referenz 0.9583*

Schema	Gitterbezeichnung	#DOF	rel. Fehler[%]	CPU-Zeit[s]
LB	4-6	48.096	1.07	10
LB	4-7	141.696	0.70	93
FEM	0+2	11.774	5.50	6
FEM	1+2	26.922	1.04	53
FEM	2+0	30.642	0.19	33
FEM	2+1	43.314	0.10	38

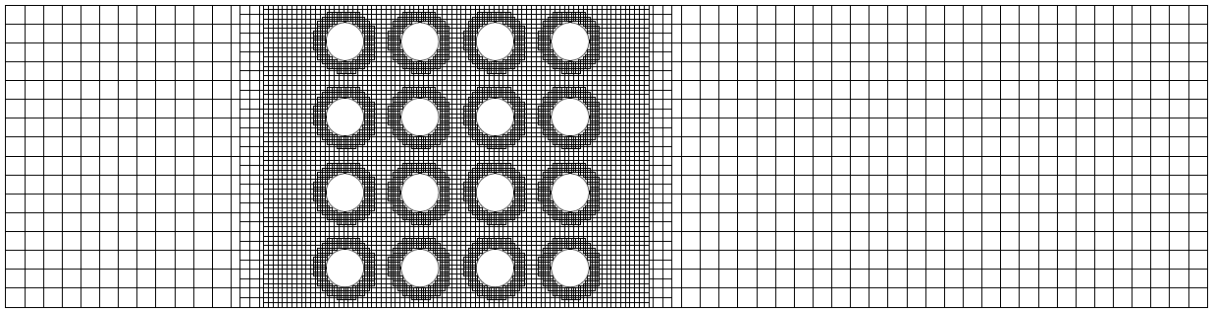


Abbildung 5.3: *Quadtree-artiges LB-Gitter (Level 4-7) mit 141.696 DOF für* $Re_E = 1$

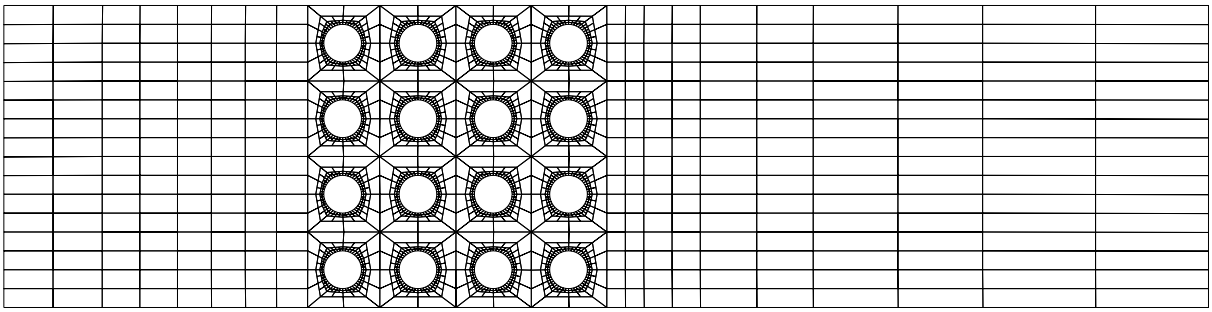


Abbildung 5.4: *Blockstrukturiertes FEM-Netz, Kennung "(1+2)" mit 26.922 DOF für* $Re_E = 1$

Der zweite Testfall ist ein transientes Strömungsproblem bei $Re_E = 200$, bei dem die Druckbeiwerte am Zylinder D berechnet werden. Es wurde eine asymmetrische Strömungsgeometrie gewählt, um sicherzustellen, dass die Strömungsasymmmetrie nicht von Netzkonfigurationsdetails oder der Löserimplementierung abhängt. Abbildung 5.6 und 5.7 zeigen zwei typische Gitter, die bei der Berechnung der Druckbeiwerte bis unter 1% Genauigkeit zugrunde lagen.

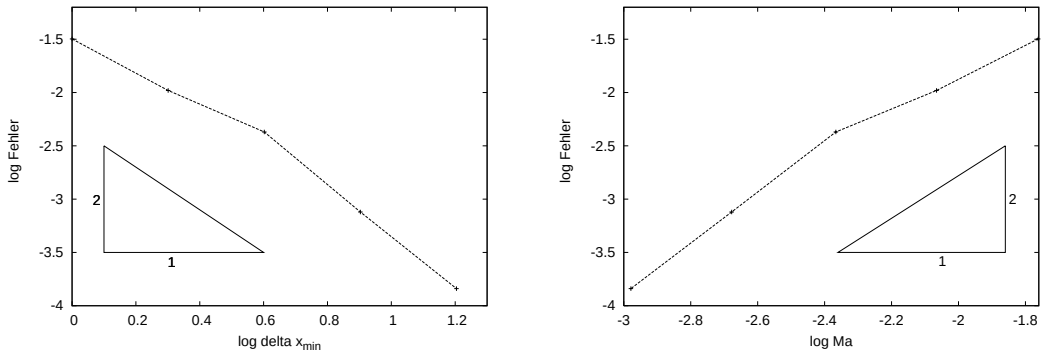
Die Referenzlösung wurde mit einer hoch aufgelösten Gitterdiskretisierung mit dem parallelen CFX-Löser berechnet. Die gemittelten Druckbeiwerte sind $c_{Drag} = 2.0548$ und $c_{Lift} = 0.9150$. Für die Pe-

Tabelle 5.3: *Konvergenz LB: Dragbeiwert, Zylinder A,* $Re_E = 1$*, Referenzwert 465,58*

Gitterbezeichnung	#DOF	Ma	c_D	rel. Fehler[%]	#Zeitschritte	Zeit[s]
4-5	15.264	0.0173	450.76	3.183	3.900	3
4-6	40.320	0.0086	470.45	1.045	2.000	10
4-7	141.696	0.0043	467.57	0.427	2.900	93
4-8	549.936	0.0021	465.94	0.075	4.900	1.140
4-9	2.182.608	0.0010	465.65	0.014	7.900	14.700
9-9	8.976.960	0.0010	465.58	0.000	152.300	45.352

Tabelle 5.4: *Konvergenz FEM: Dragbeiwert, Zylinder A,* $Re_E = 1$*, Referenzwert 465,58*

Methode	Gitterbezeichnung	#dof	c_D	rel. Fehler[%]	Zeit[s]
FEM-CC2D	(4)	217.248	461.31	0.92	24
FEM-CC2D	(5)	864.576	464.51	0.23	121
FEM-CC2D	(6)	3.449.472	465.32	0.06	493
FEM-Q2/P1	(2+4)	220.722	465.58	0.00	117
FEM-Q2/P1	(3+1)	145.794	465.14	0.10	182
FEM-Q2/P1	(3+2)	196.482	465.48	0.02	347
FEM-Q2/P1	(3+3)	297.858	465.56	0.01	573
FEM-Q2/P1	(4+1)	528.162	465.47	0.03	710
FEM-Q2/P1	(4+2)	629.538	465.56	0.01	1.297
FEM-Q2/P1	(5+0)	1.901.154	465.47	0.03	3.699
FEM-Q2/P1	(5+1)	2.002.530	465.56	0.01	3.003
FEM-Q2/P1	(5+2)	2.205.282	465.58	0.00	1.521



(a) **(b)**

Abbildung 5.5: *LB: die räumliche Konvergenz für den Dragbeiwert bei* $Re_E = 1$ *ist etwas besser als zweite Ordnung*

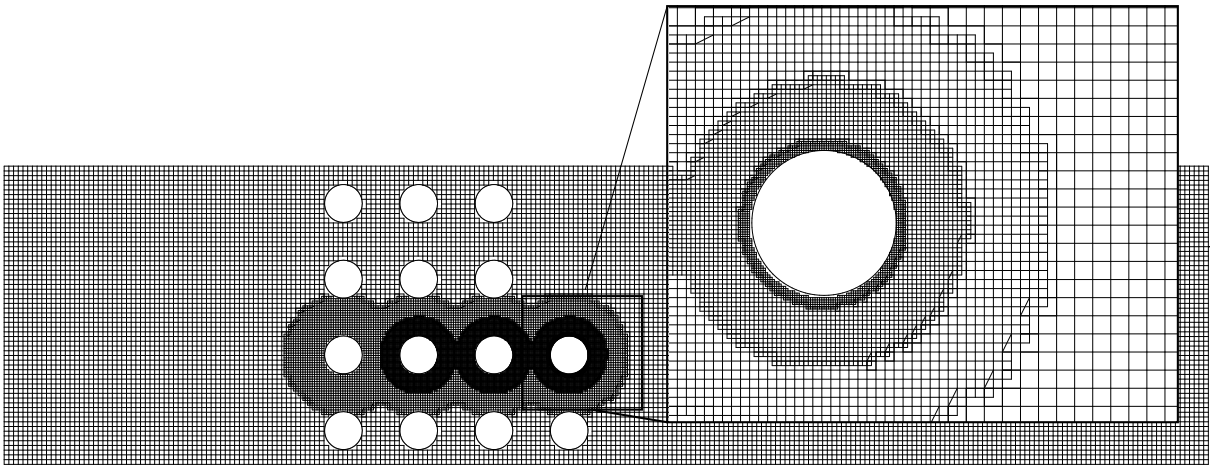


Abbildung 5.6: *Quadtree-artiges LB-Gitter (Level 6-9) mit 243.774 DOF für* $Re_E = 200$

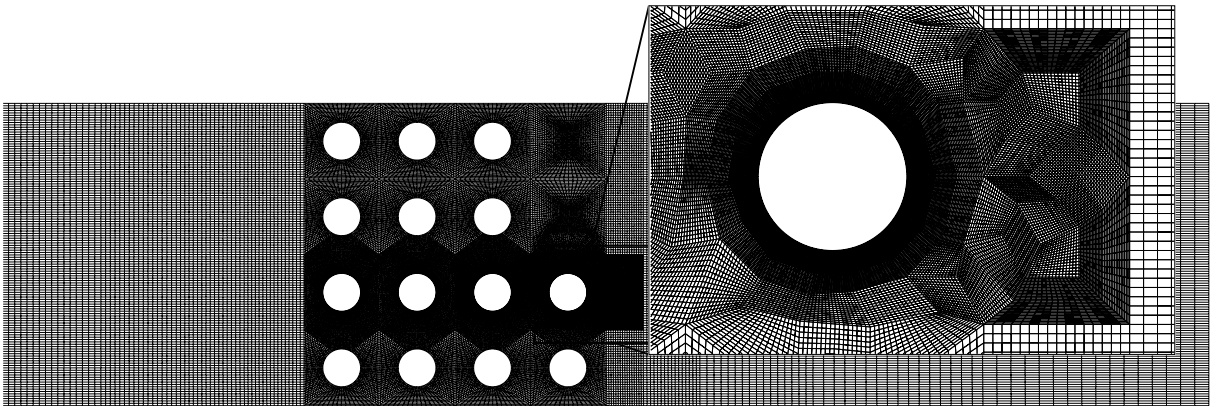
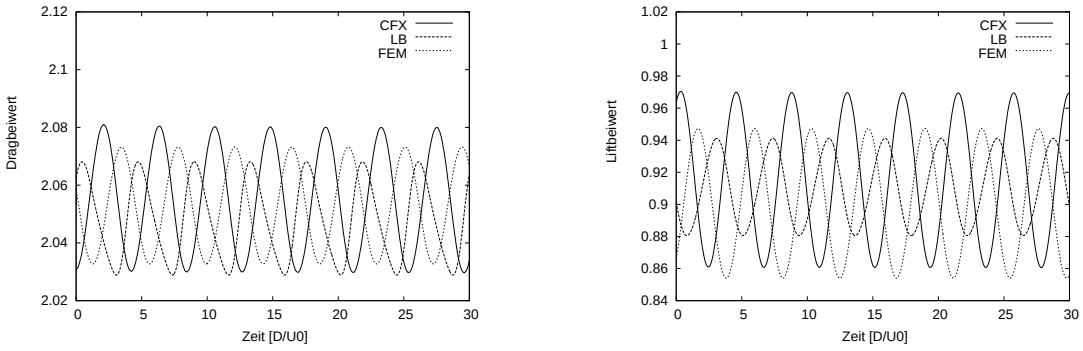


Abbildung 5.7: *blockstrukturiertes FEM-Netz, Kennzeichnung "(5)" mit 450.528 DOF für* $Re_E = 200$

riode der Dragschwingung wurde $T_{ref} = 4.2327[\frac{D}{u_0}]$ gemessen. Tabelle 5.5 zeigt die Ergebnisse der berechneten Größen für die verschiedenen Löser. Die Machzahl für die LB-Simulation beträgt Ma $= 0.02$. Es wurde das MRT-Modell verwendet, da der LBGK-Ansatz, basierend auf einem Single-Relaxation-Time-Modell, instabil für diesen Testfall ist. Abbildung 5.8 und 5.9 geben einen Eindruck für die transienten Druckbeiwerte der verschiedenen Methoden.

Tabelle 5.5: *Ergebnisse des transienten Falles (Zylinder D): Spalten 3-5 zeigen den relativen Fehler der zugehörigen Größen in %.*

Schema (Gitterbez.)	#DOF	$\Delta c_D[\%]$	$\Delta c_L[\%]$	$\Delta T_{Ref}[\%]$	CPU-Zeit[s] / T_{Ref}
LB (6-8)	199.656	2.4	0.4	0.5	30
LB (6-9)	243.774	0.3	1.6	0.5	46
FEM (4)	113.264	1.3	0.3	9.2	22
FEM (5)	450.528	0.1	1.5	0.1	265
CFX	385.485	1.6	2.5	0.2	2.856 [1]
CFX	917.616	0.5	1.3	0.2	6.594 [1]
CFX	1.807.428	0	0	0	13.440 [1]

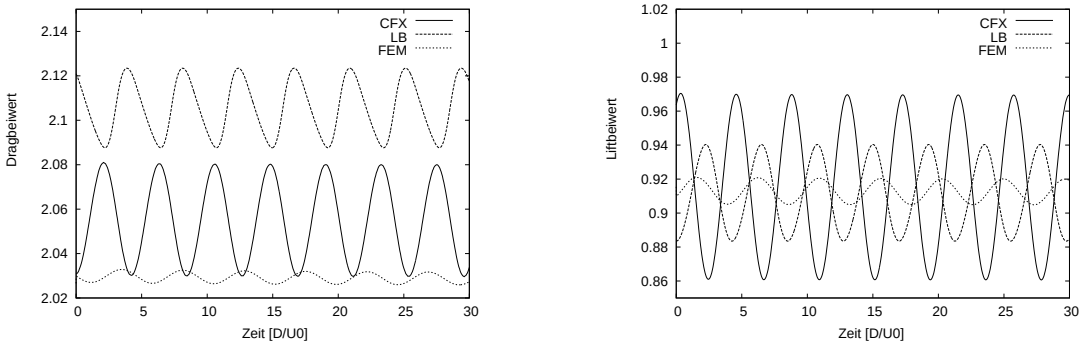


(a) Dragbeiwert **(b)** Liftbeiwert

Abbildung 5.8: *Zeitserie der Druckbeiwerte (Drag und Lift) für LBE, FEM (hohe Auflösung) und* CFX *(Referenzlösung) für* $Re_E = 200$

Die Berechnungseffizienz von CFX ist deutlich niedriger als die von VIRTUALFLUIDS. Durch optimierte Zeitschrittweiten und Netze in CFX kann diese noch erhöht werden. Aber es ist nicht davon auszugehen, dass ein Gewinn von bis zu zwei Größenordnungen erzielt wird. Es sollte beachtet werden, dass abhängig von lokalen Variationen in den entsprechenden Netzen oder Gittern die benötigte CPU-Zeit pro Periode deutlich variieren kann, die Tendenz, wie in Tabelle 5.5 gezeigt, bleibt jedoch erhalten. Die Ergebnisse der LB-Simulation zeigen eine signifikante Abhängigkeit von der gewählten Machzahl. Um diesen Effekt zu quantifizieren, wurde eine Strömungsstudie für verschiedene Machzahlen mit VIRTUALFLUIDS und CFX durchgeführt. Die Ergebnisse sind in Abbildung 5.10 dargestellt. Die Unterschiede für identische Machzahlen sind wahrscheinlich auf die verschiedenen Netze und die Unterschiede im Hinblick

[1] Das Setup der CFX-Simulation wurde nicht hinsichtlich der CPU-Zeit optimiert.



(a) Dragbeiwert **(b)** Liftbeiwert

Abbildung 5.9: *Zeitserie der Druckbeiwerte (Drag und Lift) für LBE, FEM (niedrige Auflösung) und* CFX *(Referenzlösung) für* $Re_E = 200$

der Implementierung der Randbedingungen für FV- oder LB-Simulation zurückzuführen. In diesen Simulationen hängt die Anströmkraft (Drag) erwartungsgemäß quadratisch von der Machzahl für beide Ansätze ab. Die Unterschiede zwischen den Ergebnissen der Anströmkraft für den inkompressiblen und den schwachkompressiblen Fall mit Ma $= 0.1$ können für die gemittelten Werte 10% überschreiten, wie in Abbildung 5.10 dargestellt. Zusätzlich entstehen höhere Moden. Diese typischen Machzahleffekte sollten beachtet werden, wenn man LB-Methoden für hochgenaue Berechnungen verwendet. Dies sind oftmals eine Funktion der durch die Ein- bzw. Ausflussrandbedingung induzierten Reflektionen, die durch geeignete LODI-Randbedingungen [55] reduziert werden können.

Für Strömungsprobleme mit entsprechender Machzahl Ma ≤ 0.2 auf Quadtree-artigen Gittern, ist die benötigte CPU-Zeit für die LB-Berechnung beinahe proportional zu $\frac{1}{\mathrm{Ma}}$. Im Gegensatz ist die CFX-Berechnungszeit annähernd konstant, wie man in Tabelle 5.6 [2] sehen kann.

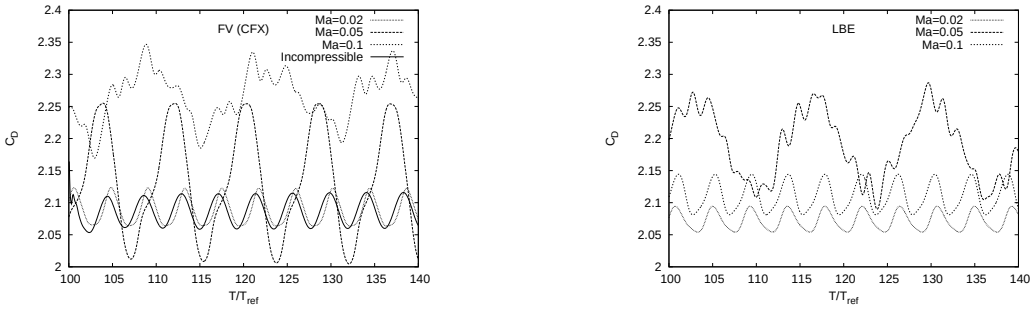
Somit kann man einen substanziellen Vorteil bei LB-Berechnung von Strömungsproblemen mit endlichen Machzahlen (Ma $\simeq 0.1$) bezüglich der Berechnungseffizienz erwarten, Der FEM-Löser kann nicht für kompressible Strömungen verwendet werden und konnte somit zur Machzahlstudie nicht beitragen.

[2] Der Unterschied der Freiheitsgrade für LB (6-8) im Vergleich in Tabelle 5.5 ist infolge unterschiedlicher Verfeinerungsbereiche

[3] Das Setup der CFX-Simulation wurde hinsichtlich der CPU-Zeit nicht optimiert

Tabelle 5.6: *CPU-Zeiten für den $Re_E = 200$ Fall Dragberechnung für verschiedene Machzahlen ($Ma = 0$ entspricht dem inkompressiblen Fall).*

Schema (Gitterbez.)	#DOF	Ma[%]	CPU-Zeit[s] / T_{ref}
LB (6-8)	318.123	0.1	12
LB (6-8)	318.123	0.05	25
LB (6-8)	318.123	0.02	61
CFX	385.485	0.1	2.730 [3]
CFX	385.485	0.05	2.814 [3]
CFX	385.485	0.02	2.772 [3]
CFX	385.485	0	2.856 [3]



(a) Finite Volumen (CFX)

(b) Lattice-Boltzmann (VIRTUALFLUIDS)

Abbildung 5.10: *Machzahlabhängigkeit der Dragergebnisse*

5.1.4 Diskussion und Ausblick

Aus theoretischer Sicht würde man erwarten, dass Multigrid-basierte Löser prinzipiell einen Vorteil gegenüber dem vergleichsweise einfachen numerischen Schema des LB-Ansatzes haben. Dieser theoretische Vorteil hat sich für zeitabhängige Testprobleme nicht bestätigt. Für den inkompressiblen Fall wurden die LB-Ergebnisse mit denen der FEATFLOW-Version mit adaptiven Zeitschrittverfahren und nichtlinearer Iteration sowie mit CFX verglichen. Es konnte vergleichbare Genauigkeit bei vergleichbaren oder niedrigeren Kosten betreffend der CPU-Zeit erhalten werden. Für Strömungen mit kleinen Machzahlen stellt sich ein substanzieller Vorteil in der Berechnungszeit für den LB-Ansatz im Vergleich mit dem Finite-Volumen basierten CFX-Code heraus. Eine interessante Untersuchung wäre, ob dieser Vorteil auch besteht, wenn Turbulenzmodelle genutzt werden.

Im Fall der stationären Strömung in komplexen Geometrien war der LB-Löser langsamer als der FEM-Ansatz. Der Basisalgorithmus der Gleichung 2.10 ist in seiner expliziten Form in Bezug auf die Berechnungskomplexität eines impliziten Ansatzes für stationäre Probleme asymptotisch unterlegen. Für kleine Reynoldszahlen wird die Navier-Stokes-Gleichung elliptischer, womit die relative Leistung eines Multigrid-Lösers für die Poisson-Gleichung ansteigt.

Das Einsatzgebiet von LB-basierten Lösern sind schwach kompressible, zeitabhängige Strömungen mit Machzahlen bis zu 0.2, da in diesem Bereich der Machzahleinfluß auf die Zeitschrittgröße mäßig ist.

Es ist irreführend zu fragen, ob LB-Methoden allgemein besser geeignet sind für CFD-Simulationen als konventionelle Methoden. Es sollte sorgfältig zwischen den Modellierungsfehlern (speziell für komplexe Fluide), Diskretisierungsfehlern und der Implementierungsart, welche im Fall von unstrukturierten Gittern beträchtlichen Einfluss auf die Berechnungseffizienz haben kann, unterschieden werden.

Die Durchführung von effizienten Simulationen benötigt die Entwicklung von adaptiven Methoden mit genauen *a posteriori* Fehlerabschätzern. Diesbezüglich sind außer heuristischen Ansätzen [18, 129] für den LB-Ansatz keine Arbeiten bekannt. Außerdem sollten Benchmarks von turbulenten Strömungen in drei Dimensionen sowie Mehrphasenströmungen durchgeführt werden.

Die Ergebnisse zeigen, dass der LB-Prototyp für die betrachteten Größen (Druckbeiwerte, Druckabfall) konkurrenzfähig für inkompressible transiente Probleme ist. Für stationäre Stokesströmungen ist er asymptotisch langsamer, da die algorithmisch-asymptotische Komplexität der klassischen LB-Methode im Vergleich mit Multigrid-Lösern, welche in dem FEM- und FV-Code integriert sind, nicht optimal ist.

Für schwach kompressible Fälle hat der LB-Ansatz verglichen mit CFX einen signifikanten Zeitvorteil. Dies bedeutet, dass der LB-Löser extrem effizient im Bereich von schwach kompressiblen Strömungen ist. Es wurde außerdem gezeigt, dass der Einfluß der finiten Machzahl in LB-Simulationen für Lösungen der inkompressiblen Navier-Stokes-Gleichung nicht zu vernachlässigen ist.

5.2 Oszillierender Zylinder

Mittal und Kumar prüften in [88] sehr ausführlich wirbelinduzierte Schwingungen eines Zylinders in einer Strömung mit Re=325. Sie nutzten hierbei eine Raum-Zeit-Finite-Element-Formulierung. Der Zylinder ist als Einmassenschwinger mit zwei translatorischen Freiheitsgraden formuliert. Es werden Systeme mit verschiedenen Eigenfrequenzen im Verhältnis zur Strouhal-Zahl des stationären Zylinders untersucht. Hierbei wird die Masse des Zylinders konstant gehalten und die Federkonstante variiert.

Bei Reynoldszahlen größer 100 wirken wechselnde Kräfte durch Wirbelablösung auf den Zylinder. Diese Kräfte bewirken eine Beschleunigung und somit die Bewegung des Zylinders. Die Zylinderbewegung hat wiederum Einfluß auf das umgebende Fluid. Es kommt zu einer bidirektionalen Fluid-Struktur-Wechselwirkung.

Ist die Eigenfrequenz kleiner als die Anregefrequenz stellt sich die Wirbelablösung darauf ein. Man spricht vom sogenannten *Lock-in*-Effekt.

Diese Testfälle wurden bereits von Leydag [76] mit dem Lattice-Boltzmann-Ansatz auf uniformen Gittern nachgerechnet. Die Ergebnisse stimmen gut mit denen von Mittal und Kumar überein.

Da sich dieser Testfall gut zur Validierung eignet, soll er hier mit aktuellen Ergebnissen aus dem nichtuniformen Strömungssimulator gezeigt werden.

Das Setup für die Berechnung wird über dimensionslose Parameter eingestellt. Die Eingangsgrößen zur Dimensionsanalyse sind die Dichte ρ, die Geschwindigkeit $\bar{u}$ und die Viskosität ν des Fluids sowie der Durchmesser D, die Masse m, die Steifigkeit k und die Dämpfung c der Struktur (des Einmassenschwingers).

Reynoldszahl	Re	$=$	$\frac{\bar{u}\,D}{\nu}$
dimensionslose Masse des Zylinders	$\tilde{m}$	$=$	$\frac{4\,m}{D^2\,\rho}$
dimensionslose Eigenfrequenz der Federn	F_s	$=$	$\frac{D\sqrt{k/m}}{2\,\pi\,\bar{u}}$
Dämpfungsmaß (Lehrsches Dämpfungsmaß)	ζ	$=$	$\frac{c}{2\,m\sqrt{k/m}}$

Die Geometrie des Berechnungskanals wird mit folgenden Werten definiert: Durchmesser D des Zylinders, Länge des System $L = 30.5D$ und die Höhe $H = 16D$. Der Zylinder befindet sich an der Position $x = 8D$ und $y = 8D$ in Ruhe. Die physikalischen Größen sind: $Re = 325$, $\tilde{m} = 4.7273$ und $\zeta = 0.00033$. Die bezogene Eigenfrequenz wird variiert. In den Abbildungen 5.12,5.13 und 5.14 werden die Ergebnisse für $F_s = 0.21, F_s = 0.42$ und $F_s = 0.53$ gezeigt, die mit den Referenzkurven übereinstimmen.

Das Basisgitter mit 610 x 320 Knoten wurde im oszillierenden Bereich a priori verfeinert, wie in Abbildung 5.11 für zwei Verfeinerungsstufen zu erkennen ist. Damit ergibt sich ein Zylinderdurchmesser von 80 LB-Knoten auf dem feinsten Level. Dies führt zu einer Elementreynoldszahl von $Re_{element} > 16.25$. Je nach Geschwindigkeit des Zylinders kann dieser Wert deutlich überschritten werden. Die dimensionslose Masse ist in diesem Beispiel relativ klein, wodurch Probleme auftreten können und sich infolge dessen kein periodischer Zustand einstellen kann. Um eine bezogene Eigenfrequenz von $F_s = 0.105$ zu erhalten muss die Federkonstante sehr klein gewählt werden. Dadurch wird das System sensibler als

es aufgrund der hohen Elementreynoldszahl und der geringen Masse ohnehin schon ist. Somit wird für diesen Fall kein brauchbares Ergebnis erzielt, da sich kein periodischer Zustand einstellt.

Die übrigen Berechnungen hingegen liefern gute Übereinstimmungen, wie in den Abbildungen 5.12,5.13 und 5.14 dargestellt. Die Auslenkungen sind nahezu deckungsgleich. Für den Fall, dass die Erregerfrequenz gleich der Eigenfrequenz ($F_s = 0.21$) ist, sind die Auslenkungen erwartungsgemäß deutlich größer als in den anderen Berechnungen. Für Bauwerke hat dies eine sehr große Bedeutung, da große Auslenkungen zu starken Beschädigungen führen können.

Betrachtet man die Kraftverläufe, sind deutlich Unstetigkeiten im Kraftverlauf zu erkennen. Diese entstehen wenn ein Knoten den Status von Solid zu Fluid und ungekehrt wechselt. In [76] wurden verschiedene Interpolationsalgorithmen zur Vermeidung dieser Oszillationen getestet: (1.) Initialisierung durch Ansetzen der Gleichgewichtsmomente, (2.) lineare Extrapolation der Momente aus dem Strömungsfeld, (3.) quadratische Extrapolation der Momente aus dem Strömungsfeld und (4.) Ermittlung der Verteilungen durch eine Shepard-Interpolation [112]. Hier wird der in Kapitel 2.2.1 beschriebene Algorithmus mit einer Poisson-artigen Iteration benutzt, um den neu entstandenen Fluidknoten zu initialisieren.

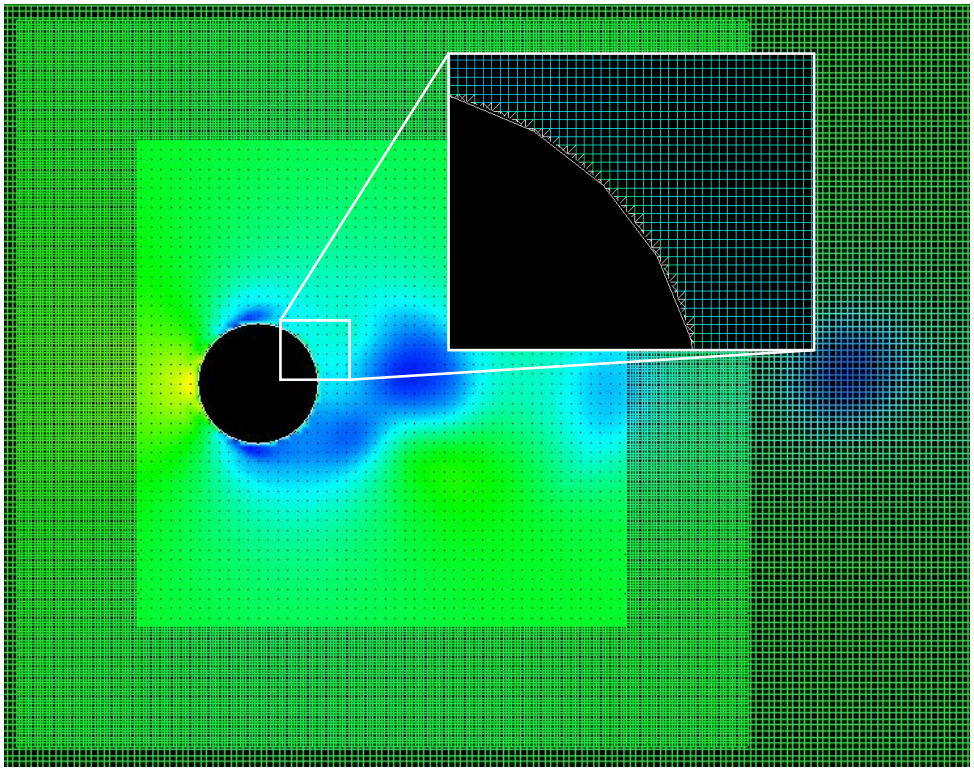
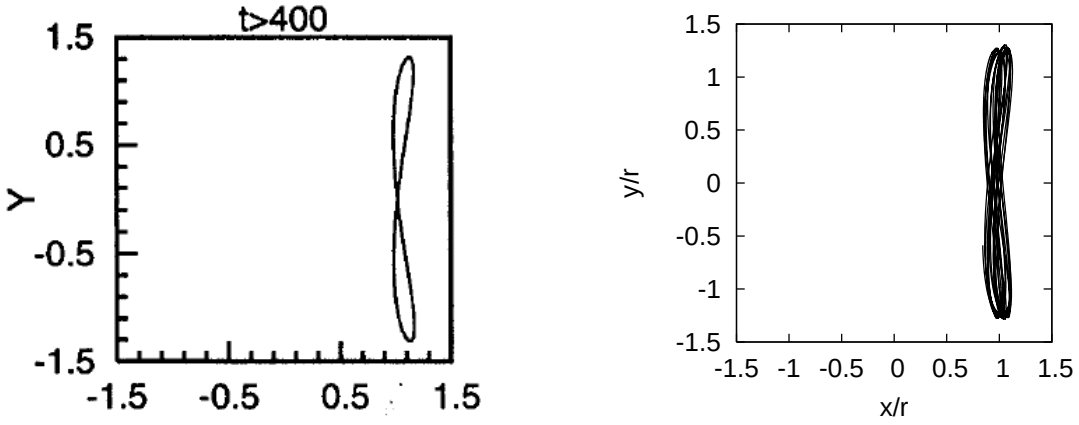
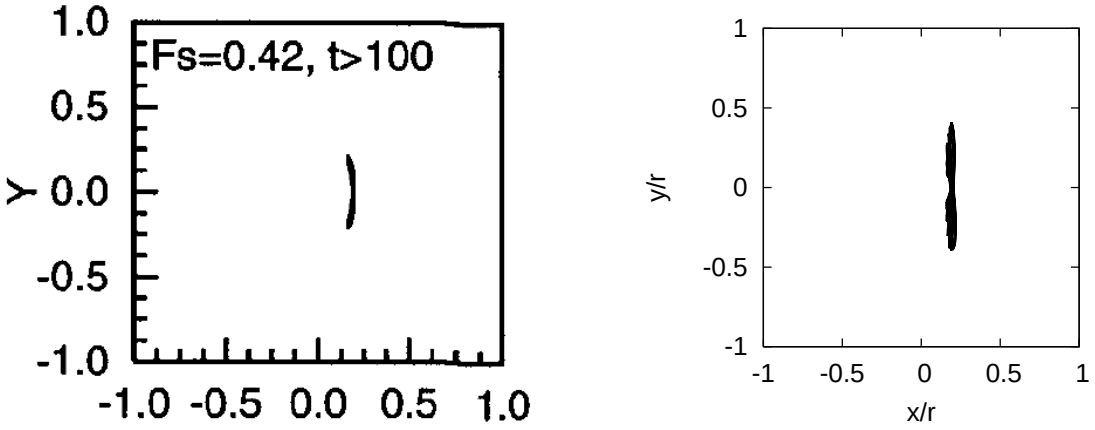


Abbildung 5.11: *Gitterverfeinerung im Bereich des oszillierenden Zylinders*



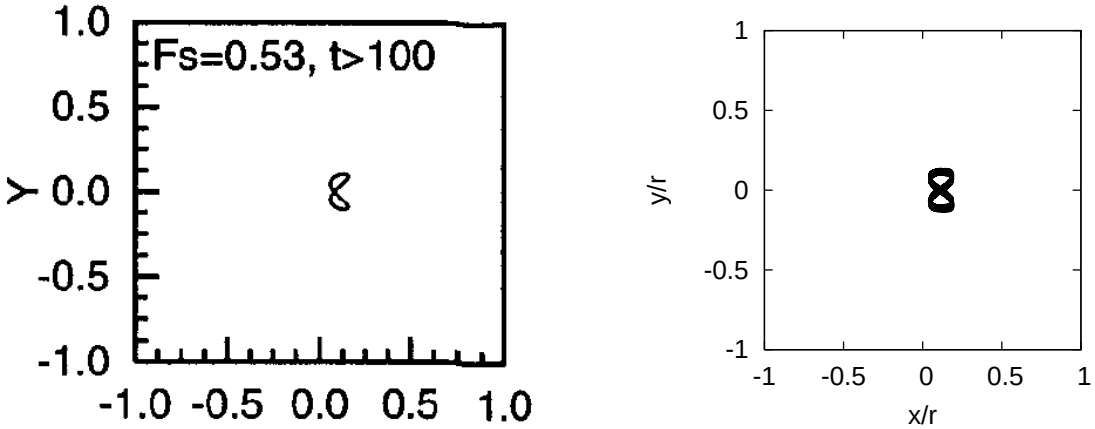
(a) Literaturwerte aus [88] **(b)** VIRTUALFLUIDS

Abbildung 5.12: *Auslenkung für* $F_s = 0.21$



(a) Literaturwerte aus [88] **(b)** VIRTUALFLUIDS

Abbildung 5.13: *Auslenkung für* $F_s = 0.42$



(a) Literaturwerte aus [88] **(b)** VIRTUALFLUIDS

Abbildung 5.14: *Auslenkung für* $F_s = 0.53$

5.3 Der numerische Benchmark

Die detailierte Beschreibung des FSI-Benchmarks, der von der DFG-Forschergruppe 493 definiert wurde, kann im seperaten Beitrag von Turek & Hron [51] nachgelesen werden. Wie in Abbildung 5.15 dargestellt, befindet sich bei diesem Testfall ein Zylinder mit einem angehängten elastischen Kragarm asymmetrisch in einem Kanal.

Der Benchmark besteht aus drei verschiedenen Teilen,

- einem reinen Fluidbenchmark (CFD),
- einem reinen Strukturbenchmark (CSM) und
- einem gekoppelten Benchmark (FSI).

Die Reynoldszahlen 20, 100 und 200 werden berechnet. Die CFD- und CSM-Testfälle dienen der Validierung der einzelnen Löser.

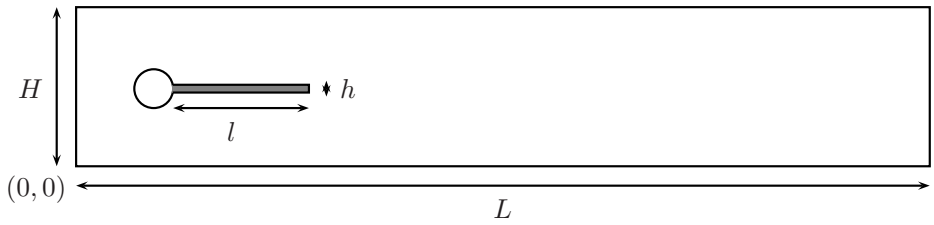


Abbildung 5.15: *Berechnungsgebiet*

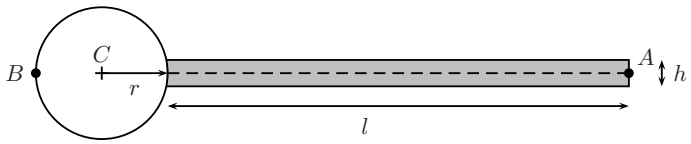


Abbildung 5.16: *Detailansicht der Struktur*

Die geometrisch definierten Parameter sind:

- Kanallänge $L = 2.5\mathrm{m}$
- Kanalhöhe $H = 0.41\mathrm{m}$
- Kreismittelpunkt $C = (0.2\mathrm{m}, 0.2\mathrm{m})$
- Kreisradius $r = 0.05\mathrm{m}$
- Länge des elastischen Balkens $l = 0.35\mathrm{m}$
- Höhe des elastischen Balkens $h = 0.02\mathrm{m}$
- Kontrollpunkt $A = (0.6\mathrm{m}, 0.2\mathrm{m})$

Am Kontrollpunkt A werden die Verschiebungen gemessen. Die physikalischen Parameter werden später in den Unterkapiteln gegeben. Haftrandbedingungen (no-slip) werden am oberen und unteren Rand gesetzt.

Das Newtonsche Fluid hat eine Dichte von $\rho^f = 10^3[\frac{kg}{m^3}]$ und eine kinematische Viskosität von $\nu^f = 10^{-3}[\frac{m^2}{s}]$. Die Einflußgeschwindigkeit ist parabolisch mit einer Hauptgeschwindigkeit $\bar{U}$ von $0.2[\frac{m}{s}]$, $1.0[\frac{m}{s}]$ oder $2.0[\frac{m}{s}]$, so dass die drei verschiedenen Teilbenchmarks FSI1, FSI2 and FSI3 mit den jeweiligen Reynoldszahlen 20, 100 and 200 resultieren. Die Dichte der Fahne ist $\rho^s = 10^3[\frac{kg}{m^3}]$ für die Fälle FSI1 und FSI3 sowie $\varrho^s = 10^4[\frac{kg}{m^3}]$ für den Fall FSI2. Die Elastizitätsmodule des St. Venant-Kirchhoff Materials sind $E^s = 5.6 \cdot 10^6[\frac{N}{m^2}]$ für den Fall FSI3 und $E^s = 1.4 \cdot 10^6[\frac{N}{m^2}]$ für FSI1 und FSI2.

Für die eindimensionalen Balkenelemente ist die Geometrie für den Balken gemäß Abbildung 5.17 konfiguriert. Die Balkenelemente sind in der Mitte des Balkens positioniert. Die Lasten werden von den Knoten des Randes (markiert mit ungefüllten Kreisen) zu den Knoten der Balkenelemente (markiert mit gefüllten Kreisen) übertragen. Die Verschiebung des Balkens wird für die gefüllten Knoten berechnet. Die Werte werden durch einfache geometrische Zusammenhänge für die Randknoten berechnet. Die Länge l_{bar} des Balkens ist $l_{bar} = l + l^{add} = l + 0.00102\text{m}$.

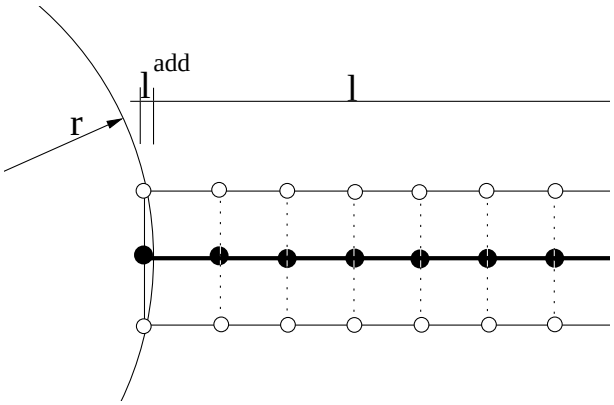


Abbildung 5.17: *Detailansicht der 1-D Balkenkonfiguration*

5.3.1 Benchmark Teil 1: CFD

Am Einlass ist ein parabolisches Geschwindigkeitsprofil und am Auslass ein konstanter Referenzdruck vorgegeben. Die Reynoldszahl ist mit $Re = \frac{\bar{U}D}{\nu_f}$ definiert, wobei $\bar{U}$ der Mittelwert der Einlaufgeschwindigkeit und D der Durchmesser des Zylinders ist. Tabelle 5.7 zeigt eine gute Übereinstimmung für die Testfälle $Re = 20$ und $Re = 100$. Der Fehler im Hinblick auf die Referenzwerte in [51] liegt unter einem Prozent für die Druckbeiwerte des Hindernisses. Der Testfall $Re = 200$ ist eine instationäre Strömung mit periodischen Schwingungen. Für diesen Fall ist eine Abweichung von unter 3% für den gemittelten Dragwert und von ca. 15% für die Amplitude der Schwingung ermittelt worden. Für den Lift ist die Abweichung 30% für den gemittelten Wert und unter 10% für die Amplitude der Schwingung.

Für stationäre Probleme wurden Fehlerindikatoren, wie z. B. die Geschwindigkeitsdivergenz, genutzt, um das Gitter adaptiv zu verfeinern (siehe Abbildung 5.18). Die Ergebnisse für den Testfall CFD3 (Re=200) sind sehr sensibel gegenüber dem Berechnungsgitter, was auch von den anderen teilnehmenden Gruppen beobachtet wurde. Die Simulationen wurden mit einer Machzahl von $Ma = 0.02$ durchgeführt.

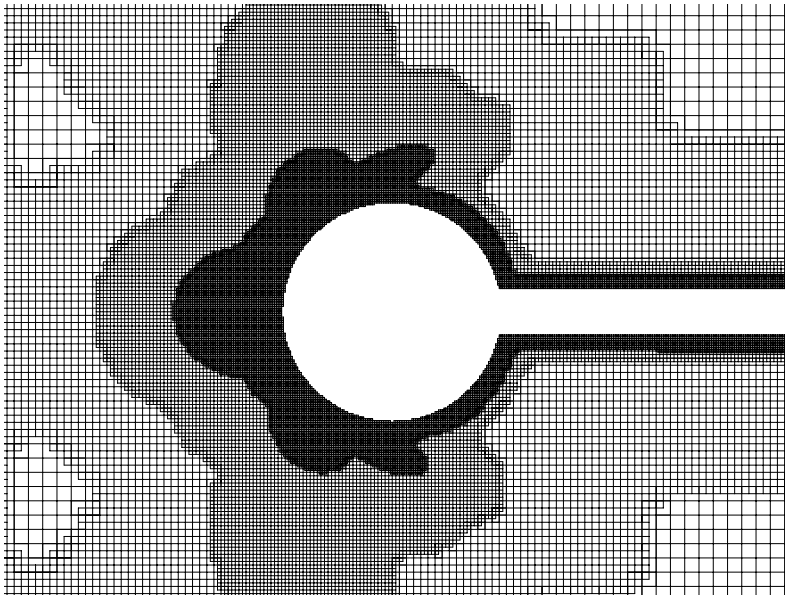


Abbildung 5.18: *adaptiv verfeinertes Gitter für* $Re = 20$

Tabelle 5.7: *CFD Benchmarkergebnisse*

Benchmark	Featflow	LB, Level 6–7	LB, Level 6–8	LB, Level 6–9
CFD1, Re 20				
– Drag in [N]	14.29	14.224	14.264	14.267
– Lift in [N]	1.119	1.098	1.108	1.116
– #DOF	11225600	406458	431082	544311
CFD2, Re 100				
– Drag in [N]	136.7	135.983	138.576	136.745
– Lift in [N]	-10.53	-11.905	-10.997	-10.587
– #DOF	11225600	771525	889524	1338660
CFD3, Re 200				
– Mittelwert Drag in [N]	439.38	459.03	456.78	450.345
– Amplitude Drag in [N]	±5.6183	±6.97	±4.4	±6.295
– Mittelwert Lift in [N]	-11.893	-15.795	-6.15	-15.47
– Amplitude Lift in [N]	±437.81	±401.035	±361.61	±466.64
– Frequenz in [Hz]	4.39	4.35	4.35	4.47
– #DOF	705152	1470978	1835811	2368944

5.3.2 Benchmark Teil 2: CSM

CSM1 und CSM2 sind stationäre Probleme mit einer konstanten Gravitationskraft $g = 2\frac{\mathrm{m}}{\mathrm{s}^2}$. Die Dynamik des Balkens in CSM3 ensteht durch eine impulsartig wirkende Gravitationskraft ($g = 2\frac{\mathrm{m}}{\mathrm{s}^2}$). Das Elastizitätsmodul (Youngsches Modul) für die Fälle CSM1 und CSM3 ist $1.4 \times 10^6 \frac{\mathrm{N}}{\mathrm{m}^2}$ und $5.6 \times 10^6 \frac{\mathrm{kg}}{\mathrm{m}^2}$ für CSM2. Für den eindimensionalen Balken wird das Elastizitätsmodul auf ein eindimensionales Elastizitätsmodul reduziert: $E_{xx} = \frac{E}{1-\nu^2}$. ν ist die Querkontraktionszahl.

Die Strukturdichte ist $\rho_s = 1000.0\frac{\mathrm{kg}}{\mathrm{m}^3}$. Die Werte in Tabelle 5.8 zeigen eine gute Übereinstimmung mit den Referenzwerten, besonders wenn man den Modellfehler im Hinblick der Annäherung der 2D-Struktur als Balken berücksichtigt. Die Anzahl der betrachteten Eigenmoden für den Fall *1D-FE-Beam Modal* war 4.

Tabelle 5.8: *CSM Benchmarkergebnisse*

Benchmark	Featflow	1D-FE-Balken Newmark	1D-FE-Balken Modal	AdhoC generalized-α
CSM1				
– y-Verschiebung in [m]	-0.06610	-0.06832	-0.06832	-0.06610
– #DOF	435776	52	52	684
CSM2				
– y-Verschiebung in [m]	-0.01697	-0.01708	-0.01708	-0.01697
– #DOF	435776	52	52	684
CSM3				
– ΔT	0.005	0.00125	0.00125	0.005
– Mittelwert Verschiebung in [m]	-0.06361	-0.06865	-0.06768	-0.06321
– Schwingungsamplitude in [m]	$\pm$0.06516	$\pm$0.06989	$\pm$0.06918	$\pm$0.06523
– Frequenz in [Hz]	1.0995	1.0702	1.0677	1.0742
– #DOF	98820	52	52	684

FEATFLOW und ADHOC lösen die Strukturgleichungen nichtlinear. Die Abweichung dieser beiden Lösungen ist marginal. Bei der eindimensionalen geometrisch linearen Approximation als Balken weichen die Werte dementsprechend ab.

5.3.3 Schwingender Balken in einer fluidgefüllten Box

Eine Teilkonfiguration des FSI-Benchmark wird studiert, um die Kopplung an einem einfacheren Setup zu testen. Es wird das Verhalten eines schwingenden Balkens in einer fluidgefüllten Box betrachtet, der mit einer Impulslastverteilung von $p = 40\frac{\mathrm{N}}{\mathrm{m}}$ beschleunigt wird. Die Fluidviskosität beträgt $\nu_f = 0.001\frac{\mathrm{m}^2}{\mathrm{s}}$ und die Fluid- sowie Strukturdichte sind jeweils $\rho_{f,s} = 1000\frac{\mathrm{kg}}{\mathrm{m}^3}$. Die stationäre Lösung ist bekannt und entspricht dem Testfall CSM1. Der zusätzliche Effekt ist die Dämpfung der Strukturbewegung durch das viskose Fluid.

Mit einer Standard-Newmark-Zeit-Diskretisierung für den Strukturlöser (1-D Balkenelemente) führt die Initialisierung bzw. Deaktivierung von Fluidknoten zu kleinen Druckstörungen, die die hohen Frequenzen der Struktur anregen. Diese interagieren mit dem Fluid und resultieren in einer Selbsterregung und einem instabilen Schema.

Wenn man eine Modalanalyse durchführt, die die Eigenwerte und Eigenvektoren des Struktursystems berechnet und nur die vier niedrigsten Eigenmoden für die Zeitintegration verwendet, ergibt sich ein stabiles Schema. Auch mit geeigneten Newmarkparametern wird das Verfahren stabil.

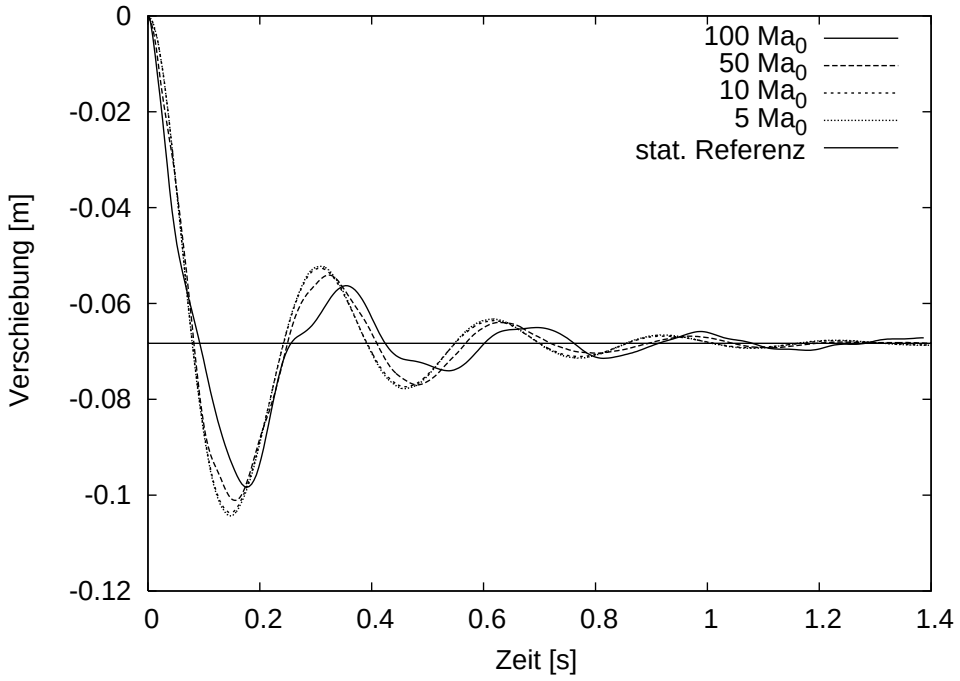


Abbildung 5.19: *gedämpfte Schwingung eines Balkens (Punkt A) in einer Box für verschiedene Machzahlen*

Abbildung 5.19 zeigt die Verschiebung des Punktes A für verschiedene Zeitschrittauflösungen, d.h. für verschiedene Machzahlen, und somit eine Konvergenz zur inkompressiblen Lösung. Die Machzahl skaliert mit $Ma = \frac{\nu_{LB}}{\nu_f} \cdot \mathrm{Ma}_0$. Ma_0 ist hierbei ein konstanter Referenzwert. In Abbildung 5.19 werden die Ergebnisse für $\nu_{LB} = 0.1, 0.05, 0.01, 0.005$, resultierend in $\Delta t_{real} = 4.307 \cdot 10^{-3}, 2.153 \cdot 10^{-3}, 4.307 \cdot 10^{-4}, 2.153 \cdot 10^{-4}$[s] (entsprechend Gleichung 4.14), gezeigt. Zur Berechnung der aktuellen Machzahl wird die Durchschnittsgeschwindigkeit von Punkt A im Intervall $[0, T_P]$ berechnet. T_P ist die Zeit, bei der die Verschiebung das erste Minimum erreicht. Diese wird durch die Schallgeschwindigkeit dividiert. Die Machzahl für den größten Zeitschritt ist verhältnismäßig hoch und hat einen Wert von annähernd 0.6, sodass Ma_0 zu 0.006 gesetzt werden konnte. In diesem Testfall wird außerdem gezeigt, dass große Verformungen (ein Viertel der Kanalhöhe) berechnet werden können.

5.3.4 FSI-Benchmark der DFG-Forschergruppe 493

Analog zu den CFD-Testfällen ist am Einlass ein parabolisches Geschwindigkeitsprofil und am Auslass ein konstanter Referenzdruck vorgegeben.

Die Konfiguration für den FSI1, FSI2 und FSI3 Benchmark sind in Tabelle 5.9 aufgelistet. Auf der Strukturseite wurde der anisotrope *Trunk Space* mit einem Ansatzgrad von sechs in Längsrichtung und vier in Dickenrichtung angewendet. Die FSI-Berechnung wurde von einer konvergierten Fluidlösung mit der fixierten Fahne gestartet. Um eine plötzliche Last, die einen Schock auf die Struktur gibt, zu vermeiden, werden die Kräfte, die auf die Struktur übertragen werden, innerhalb der ersten 300 Zeitschritte linear auf ihren wahren Wert erhöht.

Die y-Verschiebung von Punkt A wird in Tabelle 5.10 gezeigt. FSI1 resultiert in einer stationären Lösung, FSI3 hingegen in einer periodischen Lösung.

Für alle FSI-Berechnungen wurde VIRTUALFLUIDS mit dem 1-D-Balken-FEM-Code und dem Strukturlöser ADHOC gekoppelt. Die Zeitintegration für die Struktur war weder mit dem Newmark-Integrationsschema für die 1-D-Balkenelemente noch mit der Generalized-α-Methode von ADHOC stabil.

Nur die Methode der Modalanalyse und die Integration der niedrigsten Eigenmoden führte zunächst zu einem stabilen Schema für das gekoppelte Problem. Später wird gezeigt, dass durch eine geeignete Wahl der Newmarkparameter die Simulation der drei Fälle zufriedenstellend durchgeführt wurde. Der Zeitschritt für den Fluidlöser VIRTUALFLUIDS auf dem gröbsten Level und der Zeitschritt für den Strukturlöser waren identisch. Die Anzahl der betrachteten Eigenmoden war vier für den Fall *LB1* und neun für die Fälle *LB2, LB3* in Tabelle 5.10.

Die Ergebnisse für FSI1 zeigen, dass die Approximation mit 1-D-Balkenelementen nicht zur korrekten Lösung führt. Der Modellfehler ist sehr groß, da nur Kräfte senkrecht zur Achse des Balkens in Betracht gezogen werden. Hier ergibt die Kopplung mit dem Strukturlöser ADHOC gute Ergebnisse. Der Fehler für die Simulation mit der Machzahl 0.02 ist annähernd 1 % in Bezug auf die Referenzlösung von FEATFLOW.

Für den FSI3 Benchmark weisen die Ergebnisse der Simulation *LB3* einen Fehler von zirka 1 % in der Amplitude der Schwingung, ungefähr 1.5 % für die Frequenz und 3 % für die gemittelten Werte auf. Die gemittelten Werte sind annähernd 30 mal kleiner als die Schwingungsamplitude.. Auf einem 2GHz AMD OPTERON PC war für diesen Fall die Berechnungszeit fünf Stunden je Periode. Für die Fälle *LB1, LB2* betrug diese eine Stunde pro Periode. In Abbildung 5.20 sind die Oszillationen von Punkt A für FEATFLOW und VIRTUALFLUIDS gezeigt. Abbildung 5.21 zeigt Momentaufnahmen der bewegten Balkendynamik.

Tabelle 5.9: *Eingangsparameter des FSI-Benchmarks*

Parameter	FSI1	FSI2	FSI3
Dichteverhältnis β	1.0	10.0	1.0
Aerolastische Zahl Ae	35000.0	1400.0	1400.0
Reynoldszahl Re	20.0	100.0	200.0
Poissonzahl ν_s	0.4	0.4	0.4
$\rho_s = \rho_f$ in $\frac{\mathrm{kg}}{\mathrm{m}^3}$	1000	1000	1000
E in Pa	$1.4 \cdot 10^6$	$1.4 \cdot 10^6$	$5.6 \cdot 10^6$
$\bar{U}$ in $\frac{\mathrm{m}}{\mathrm{s}}$	0.2	1	2
ν_f in $\frac{\mathrm{m}^2}{\mathrm{s}}$	0.001	0.001	0.001

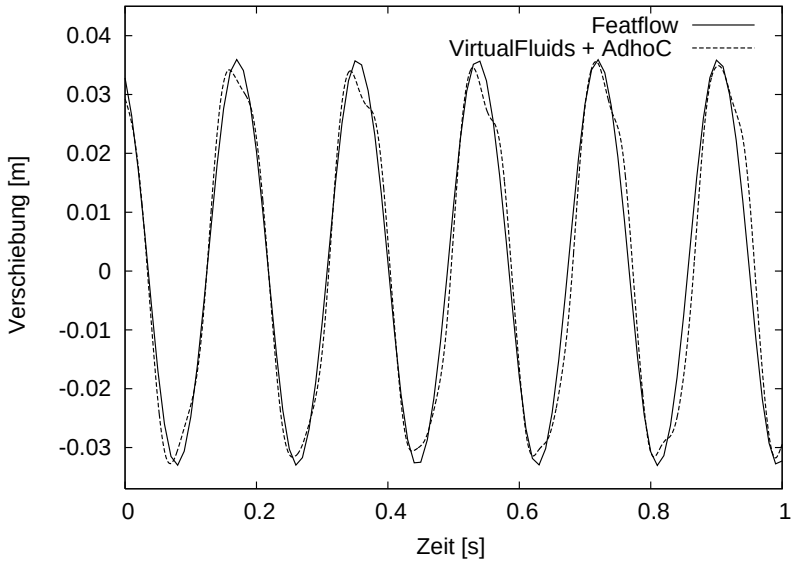


Abbildung 5.20: *Schwingung von Punkt A für* $Re = 200$

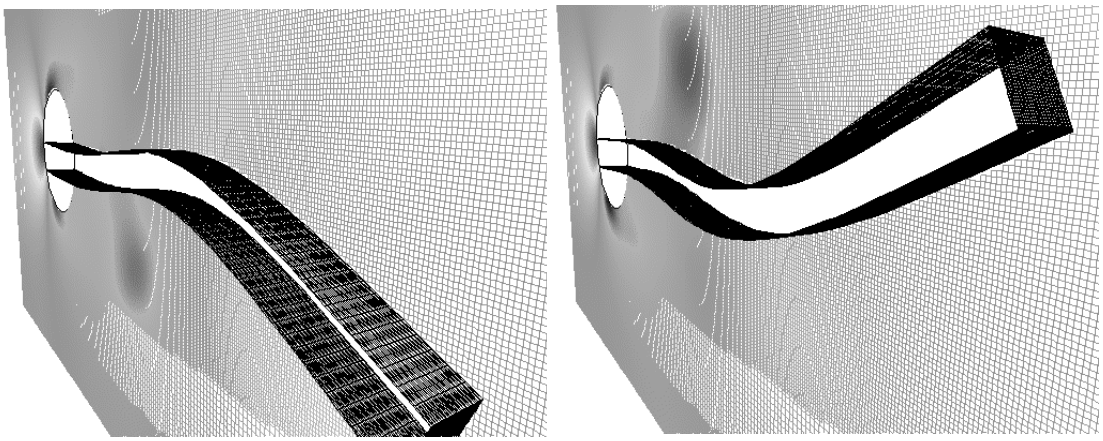


Abbildung 5.21: *Momentaufnahme des bewegten Balkens* $Re = 200$

Für die Fälle FSI1 und FSI3 liefert die Simulation mit der Modalanalyse zufriedenstellende Ergebnisse. Da die Struktur aus extrem elastischem Material zusammengesetzt ist, zeigt sie starke geometrische nichtlineare Auslenkungen für die Fälle FSI2 und FSI3. Es ist schwierig, ein stabiles Zeitintegrationsschema für solch eine Konfiguration zu definieren. Anwendungen der Generalized-α-Methode führen zu instabilen Ergebnissen in der gekoppelten Berechnung. Dies erzwingt die Nutzung des Newmarkverfahrens mit einem Parametersatz von $\beta = 0.49$, $\gamma = 0.9$, um die numerischen Instabilitäten herauszudämpfen. Aus Vergleichsgründen wurden diese Parameter auch für den stationären Fall FSI1 genutzt. Alle Fälle wurden mit einem Subcycling von zwei berechnet. Das heißt für einen Kopplungsschritt führt das Fluid zwei Zeitschritte mit internen Subiterationen aus, während die Struktur nur einen Zeitschritt ausübt. Die Machzahl des Fluides wurde für alle Fälle mit 0.1 gewählt. Die übrigen Spezifikationen sind in Tabelle 5.11 aufgelistet.

Die Auslenkungen des Fähnchens am Punkt A, dargestellt in Abbildung 5.16, sind in Tabelle 5.12 zusammengefaßt und verglichen mit den Ergebnissen, die von Hron & Turek [51] erhalten wurden. Die

Tabelle 5.10: *FSI-Benchmark: y-Verschiebung von Punkt A.*

Benchmark	Featflow	LB1 (*)	LB2(*)	LB3(*)
FSI1, Re 20				
– #dof	4835328	972054	972054	773334
– Δt [s]	–	1.263×10^{-3}	1.263×10^{-3}	0.2526×10^{-3}
– Verschiebung [m]	0.00082	0.002026	0.00076	0.00083
FSI3, Re 200				
– #dof	304128	2318904	1835863	1836495
– Δt [s]	5.0×10^{-4}	1.263×10^{-4}	1.263×10^{-4}	0.2526×10^{-4}
– Mittelwert [m]	0.00143	0.00112	0.00123	0.001386
– Amplitude [m]	$\pm$0.03437	$\pm$0.0295	$\pm$0.03988	$\pm$0.033951
– Frequenz [Hz]	5.4	5.82	5.41	5.32

(*) Legende:

- Fall LB1: Level 6-8, Ma=0.1, gekoppelt mit 1-D-Balken
- Fall LB2: Level 6-8, Ma=0.1, gekoppelt mit ADHOC
- Fall LB3: Level 6-8, Ma=0.02, gekoppelt mit ADHOC

Fall	Δt *Struktur[s]*	*LBM-Knoten*
FSI1	$2.52591e^{-3}$	[125553]
FSI2	$5.05182e^{-4}$	[160170]
FSI3	$2.52591e^{-4}$	[275646]

Tabelle 5.11: *Parameter für FSI1, FSI2 und FSI3*

Auslenkungen sind darin als $\langle Mittelwert \rangle \pm \langle Amplitude \rangle$[Hz] gegeben, wobei der $Mittelwert = \frac{1}{2}(max + min)$ und die $Amplitude = \frac{1}{2}(max - min)$ der jeweils letzten Periode der Schwingung berechnet wird. Die Frequenz wurde mit einer Fourieranalyse der periodischen Daten berechnet. Die Spalte "Zeit" wird in realer Zeit in Sekunden pro 24 Stunden auf einem 1.6GHz AMD OPTERON Prozessor berechnet.

Während FSI1 zu einer stationären Verschiebung führt, zeigen FSI2 und FSI3 eine periodische Bewegung des Fähnchens. Diese instationären Ergebnisse sind in Abbildung 5.22 und Abbildung 5.23 dargestellt, in denen auch die Ergebnisse von Hron & Turek [51] gezeigt werden.

Die Druckbeiwerte werden mit Gleichung 5.3 ausgewertet:

$$(F_d, F_l) = \int_{\partial A} \mathbf{t}\, dl \tag{5.3}$$

wo ∂A der gesamte benetzte Rand (Fähnchen und Zylinder) ist. Die Ergebnisse sind in Tabelle 5.13 zusammengefasst und mit den Ergebnissen von Hron & Turek [51] verglichen.

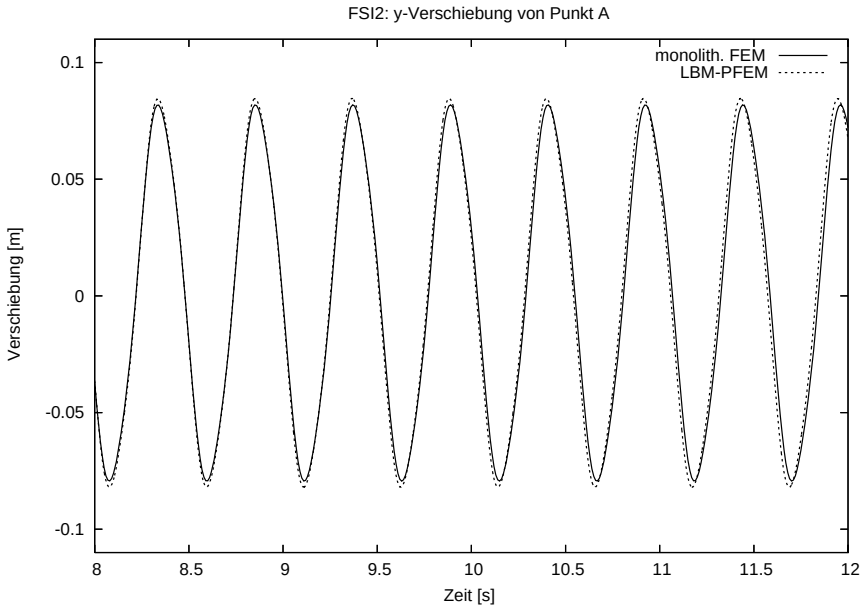


Abbildung 5.22: *FSI2: y-Auslenkung von Punkt A der LBM–p-FEM Kopplung verglichen mit [51]*

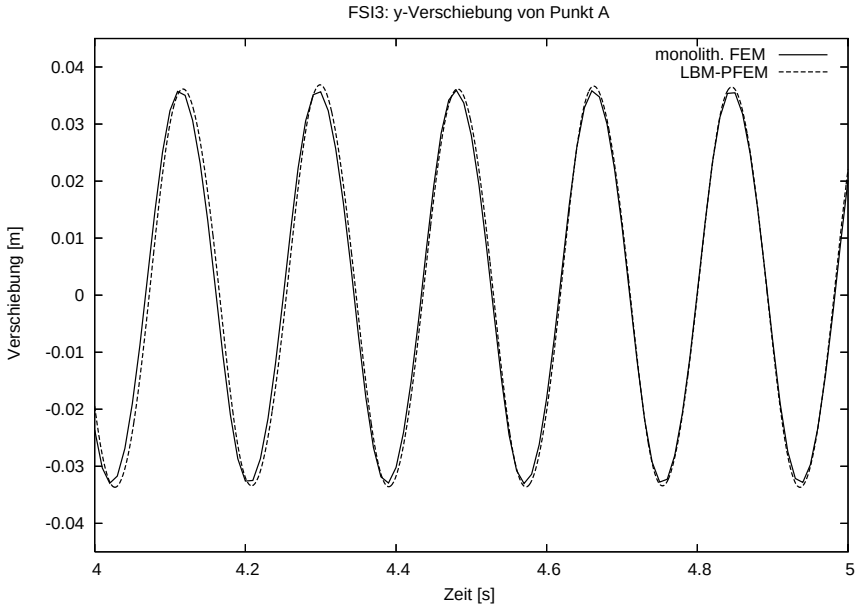


Abbildung 5.23: *FSI3: y-Auslenkung von Punkt A der LBM–p-FEM Kopplung verglichen mit [51]*

Fall	$u_x(A)[m]$	$u_y(A)[m]$	Zeit [s/Tag]
FSI1	$2.29e^{-5}$	$8.10e^{-4}$	5.2
FSI1_{ref}	$2.27e^{-5}$	$8.209e^{-4}$	$\emptyset$
FSI2	$-1.51e^{-2} \pm 1.28e^{-2}[3.8]$	$1.20e^{-3} \pm 8.34e^{-2}[1.9]$	2.7
FSI2_{ref}	$-1.458e^{-2} \pm 1.244e^{-2}[3.8]$	$1.23e^{-3} \pm 8.306e^{-2}[2.0]$	$\emptyset$
FSI3	$-2.88e^{-3} \pm 2.71e^{-3}[11.0]$	$1.48e^{-3} \pm 3.51e^{-2}[5.5]$	0.8
FSI3_{ref}	$-2.69e^{-3} \pm 2.53e^{-3}[10.9]$	$1.48e^{-3} \pm 3.438e^{-2}[5.3]$	$\emptyset$

Tabelle 5.12: *Ergebnisse und Vergleich mit der Referenz aus [51], $\emptyset$ keine Ergebnisse vorhanden*

Fall	F_l *[N]*	F_d *[N]*
FSI1	14.31	0.757
FSI1_{ref}	14.295	0.7638
FSI2	$216.60 \pm 89.31[3.8]$	$-1.23 \pm 283.00[1.9]$
FSI2_{ref}	$208.83 \pm 73.75[3.8]$	$0.88 \pm 234.20[2.0]$
FSI3	$462.53 \pm 31.34[11.0]$	$1.81 \pm 154.22[5.5]$
FSI3_{ref}	$457.3 \pm 22.66[10.9]$	$2.22 \pm 149.78[5.3]$

Tabelle 5.13: *Ergebnisse und Vergleich mit der Referenz aus[51]*

5.3.5 Schlußfolgerung

Die Ergebnisse des partitionierten Ansatzes für zwei hocheffiziente Löser (CFD und CSD) wurden detailiert beschrieben. Sie zeigen, dass der Kopplungsansatz geeignet ist und zu konsistenten, quantitativen und richtigen Resultaten führt. Das Oberflächennetz, das zuständig für den Lasttransfer ist, wird derzeit über die Gaußpunkte des p-FEM Lösers definiert. Weitere Verbesserungen des Kopplungsalgorithmus mit Hinblick auf die Robustheit werden erwartet, wenn ein Interfacenetz mit äquidistanten Gittern verwendet wird, da hier nicht die lokale Größe der Dreiecke proportional zum inversen Quadrat des Grades der Ansatzfunktion ist. Somit sind sehr kleine Größen in der Nachbarschaft von p-Elementkanten, was zu lokal reduzierter Genauigkeit im Hinblick auf den Lasttransfer führt. Durch die Einführung eines Kopplungsnetzes, das unabhängig von den Gaußpunkten mit einer passenden Projektion der Lasten vom Kopplungsnetz zum Strukturlöser und einer globalen Fehlerminimierung in Bezug auf den Transfer der Verschiebungen vom Strukturlöser zum Kopplungsnetz ist, wird eine deutliche Reduzierung dieser Effekte erwartet. Dies führt für Fälle, bei denen die höheren Moden essenziell für das gekoppelte Problem werden, zu einer verbesserten Stabilität

Außerdem ist zu erwähnen, dass zusätzlich ein impliziter Kopplungsalgorithmus getestet wurde, der ein quantitatives Gleichgewicht der Lasten und Verschiebungen jeweils auf der Fluid- und Strukturseite ermöglicht. Die Lösungen waren hier sehr stark gedämpft, so dass sich die implizite Kopplung als nicht konsistent herausstellte.

5.4 Der experimentelle Benchmark

Als weiterer Testfall zur Validierung der Fluid-Struktur-Wechselwirkung wurde in der DFG Forschergruppe 493 ein experimentelles Setup definiert. Hierbei handelt es sich ebenfalls um die Strömung in einem Kanal um ein Balken mit Frontzylinder. Im Gegensatz zum numerischen Benchmark ist das Fähnchen extrem dünn und eine zusätzliche Masse befindet sich an dessen Ende. Der Frontzylinder ist außerdem rotierend gelagert. Untersucht werden hier die Fälle $Re = 140$ und $Re = 190$, die den ersten beiden Eigenmoden der Schwingung entsprechen.

Die Dichte des Materials der Struktur beträgt für den Frontkörper aus Aluminium $2828 kg/m^3$, für die Membran und die hintere Masse aus Edelstahl $7855 kg/m^3$. Das E-Modul beträgt $200 kN/mm^2$.

Als Fluid wird Polyethyleneglycolsirup im Experiment verwendet. Es hat eine Dichte von $1050 kg/m^3$ und eine kinematische Viscosität von $0.000164 m^2/s$.

Der Frontzylinder befindet sich mittig im Abstand von $55mm$ vom Einlass entfernt (Abbildung 5.25). Das Koordinatensystem hat seinen Ursprung im Mittelpunkt des Zylinders. Der Kanal hat eine Länge von $338mm$ und eine Höhe von $240mm$. Die Gravitation wirkt entlang der x-Achse.

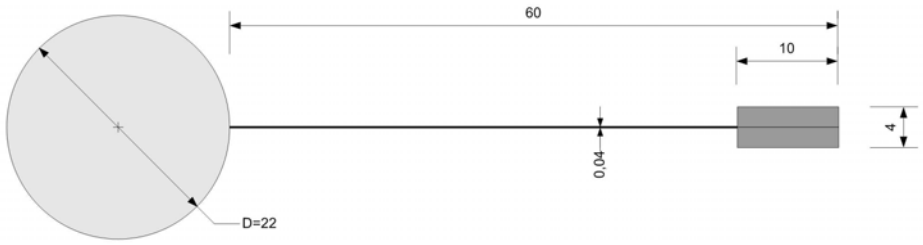


Abbildung 5.24: *Geometriedefinition der Struktur in mm [42]*

Die Erdbeschleunigung von $9.81 \frac{m}{s^2}$ ist entsprechend Gleichung 4.18 für das LB-System skaliert. Die Steifigkeit (EI) des Fähnchens beträgt $1/1000$ und die Masse $1/700$ im Vergleich zum numerischen Benchmark. Dies führt zu deutlich instabilerem Verhalten.

Auf groben LB-Gittern konnte nur eine stabile Simulation erreicht werden, wenn man die Masse der Membran erhöht hat. Das heisst, für ein Fluidgitter mit 340 mal 240 Knoten auf dem gröbsten Level wurde die Masse der Membran vervierfacht. Für ein Gitter mit 680 mal 480 Knoten wurde sie verdoppelt und für ein hochaufgelöstes Rechengebiet mit 1360 auf 960 Knoten konnte die Berechnung ohne Massenerhöhung durchgeführt werden. Die Simulationszeit betrug für das grobe Gitter mit 240 Knoten Höhe 100 Stunden für 1.6 Sekunden Echtzeit. Für das Gitter mit 480 Konten Höhe wurden 245 Stunden Rechenzeit für die 1.6 Sekunden an Echtzeit auf einem 2.0 GHz AMD Opteron 64-Bit Prozessor benötigt. Die Machzahl betrug $Ma = 0.05$.

Die Schwingungsfrequenzen für die ersten beiden Eigenformen sind im Experiment 6.38 Hz und 13.58 Hz. Durch die gekoppelte Simulation mit dem LB-Fluidlöser VIRTUALFLUIDS und dem Strukturlöser ADHOC wurden Frequenzen von 6.71 Hz für die erste Form und 16.7 Hz für die zweite Form bestimmt. Die Trajektorien des Endpunktes der Membran sind in Abbildung 5.26 und Abbildung 5.27 dargestellt.

Andere Verfahren, wie z. B. das ALE-Schema, haben bei diesen Auslenkungen starke Probleme mit der Gitterkonfiguration beziehungsweise werden sehr aufwändig beim Neuvernetzungsprozess. In Ab-

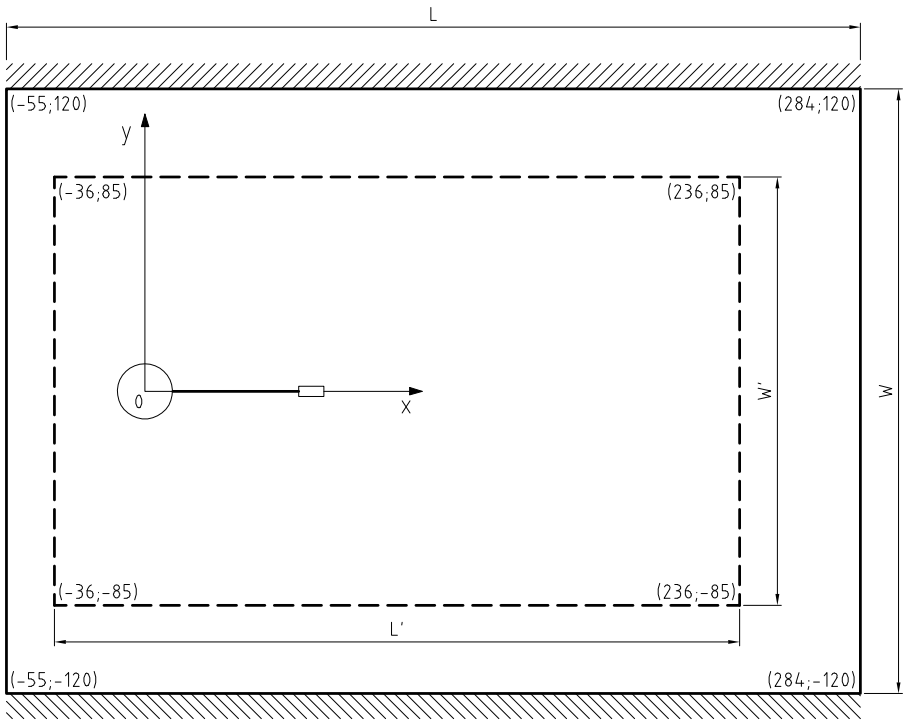


Abbildung 5.25: *Physikalisches Feld mit Meßbereich (gestrichelt) in mm [42]*

bildung 5.28 und Abbildung 5.29 werden Momentaufnahmen einer Oszillation verglichen mit dem Experiment gezeigt.

Obwohl die Ergebnisse qualitativ zufriedenstellend sind, ist der Fehler in der Frequenz mit 20% recht hoch und benötigt weitere Untersuchungen. Hier können weitere Analysen mit geänderten Kopplungsschemen hilfreich sein.

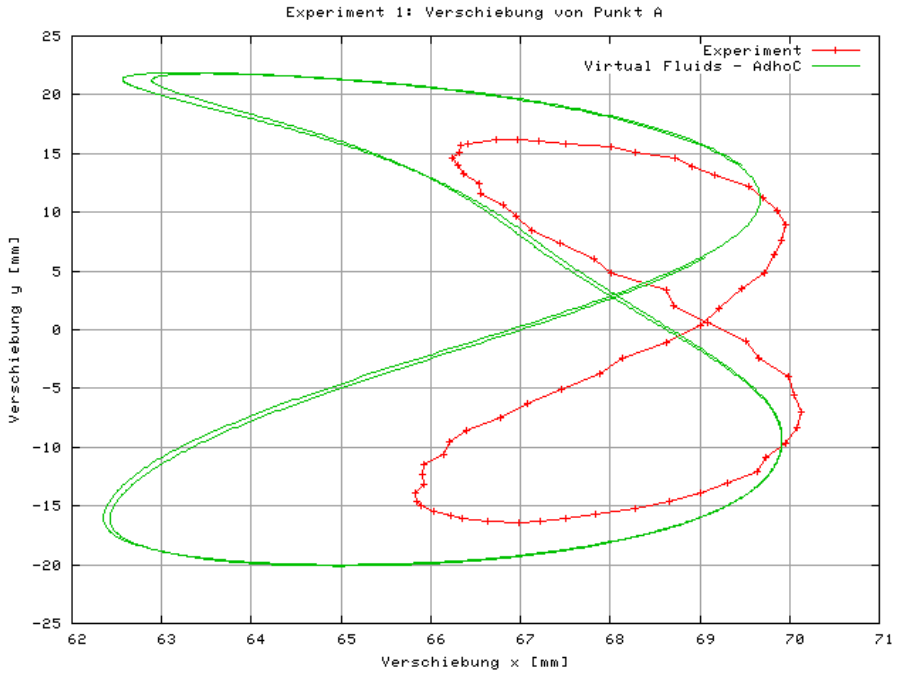


Abbildung 5.26: *Trajektorien des Endpunktes verglichen mit Messergebnissen für* $Re = 140$

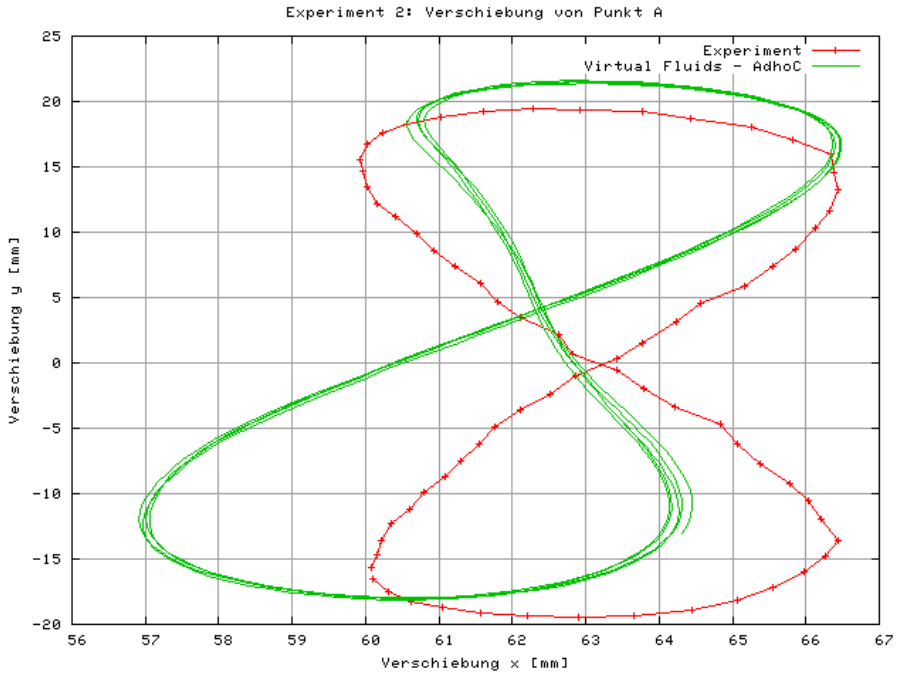


Abbildung 5.27: *Trajektorien des Endpunktes verglichen mit Messergebnissen für* $Re = 190$

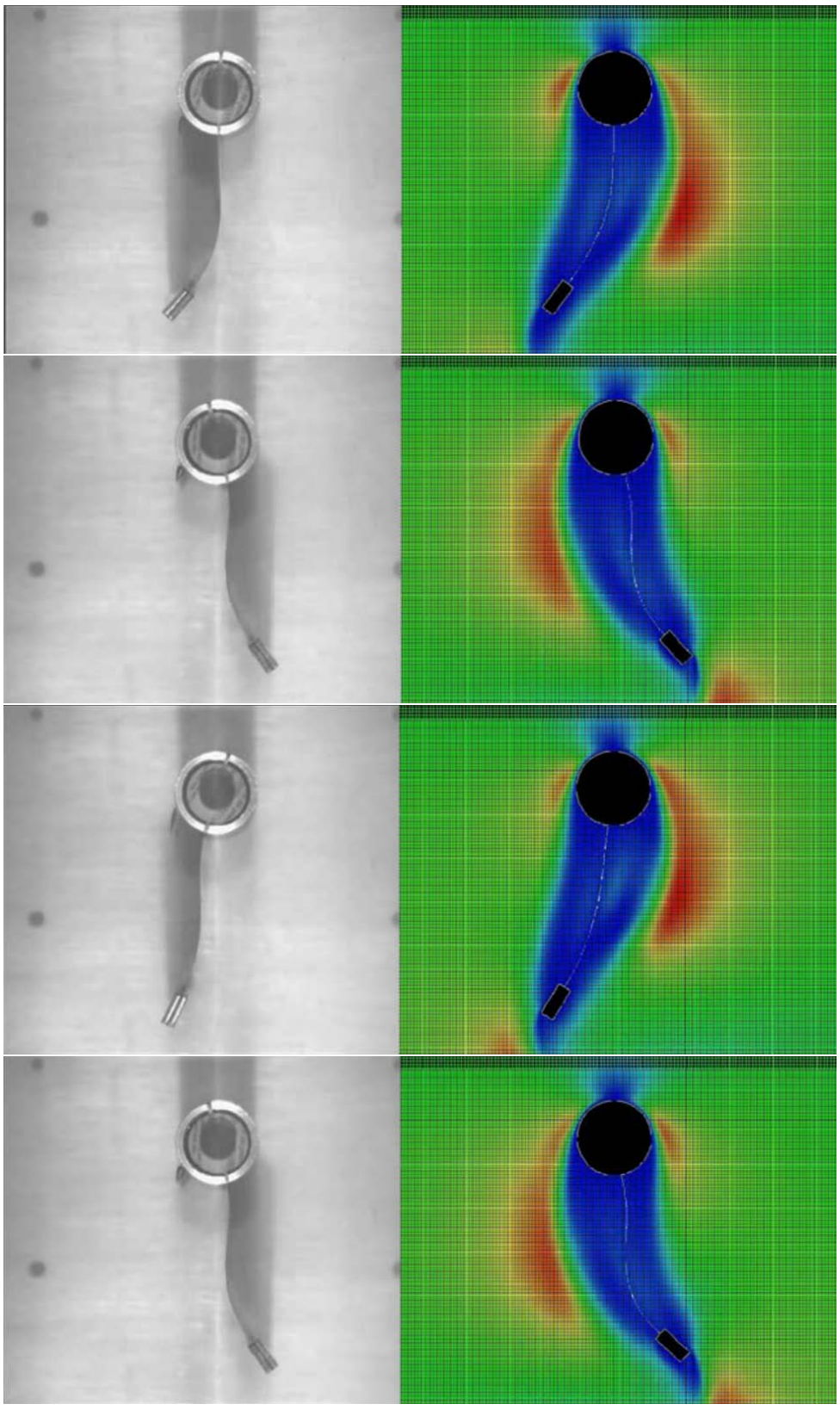


Abbildung 5.28: *Zeitreihe der numerischen und experimentellen Simulation für* $Re = 140$

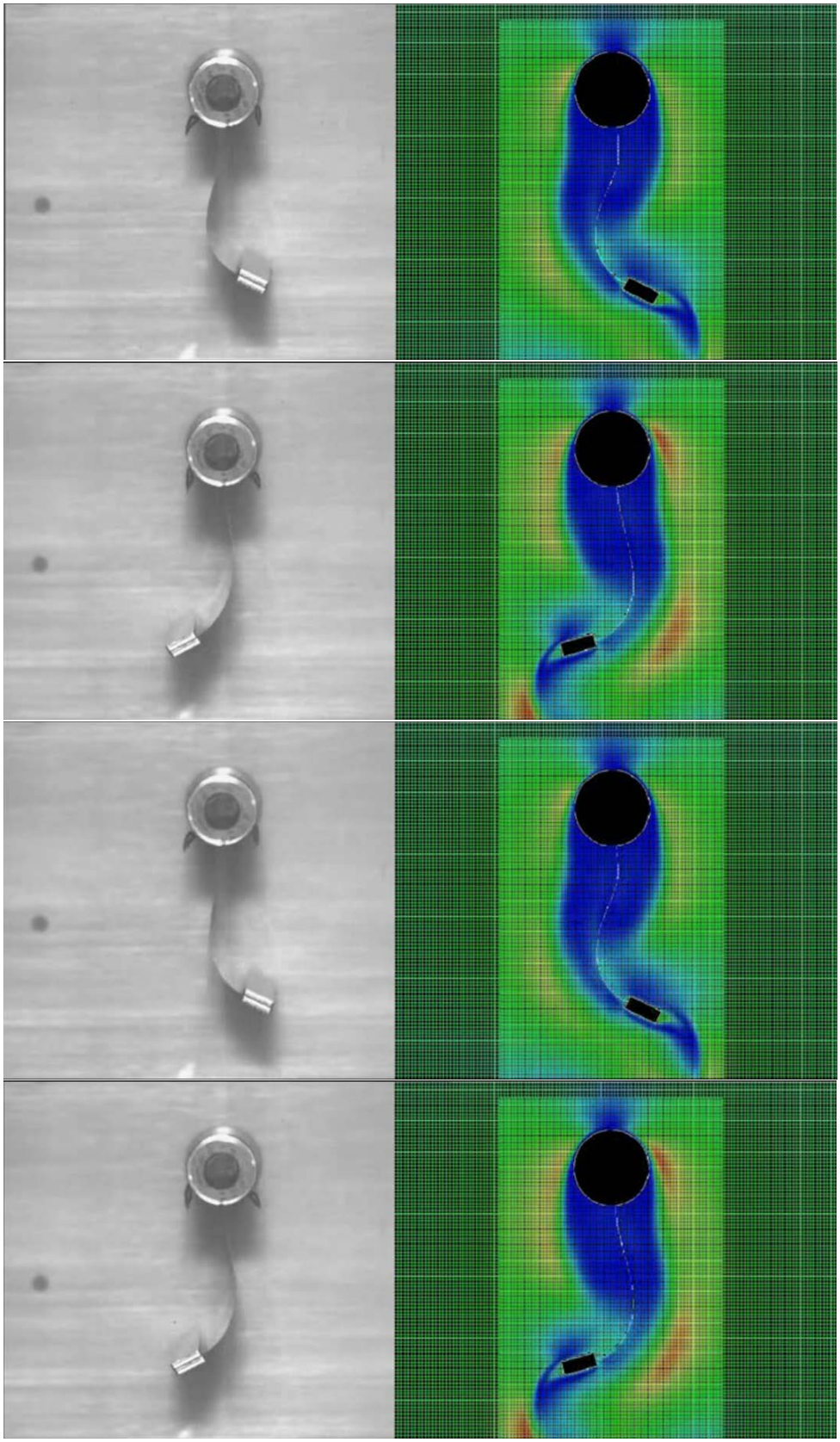


Abbildung 5.29: *Zeitreihe der numerischen und experimentellen Simulation für* $Re = 190$

5.5 Zusammenfassung

Der Prototyp des zweidimensionalen Simulationscodes VIRTUALFLUIDS wurde einer ausführlichen Validierung unterzogen. Der Benchmark für poröse Medien zeigt, dass das LB-Verfahren einen wichtigen Stellenwert unter den Fluidlösern einnimmt. Es wurden sehr effizient numerisch korrekte Ergebnisse erzielt. Beim oszillierenden Zylinder wurden die Trajektorien aus der Referenzliteratur getroffen. Am ausgiebigsten wurde der numerische Benchmark von Kapitel 5.3 validiert und liefert auch für geometrisch nichtlineare Fälle sehr gute Ergebnisse. Erfahrungen aus der DFG-Forschergruppe 493 zeigen, dass vergleichbare Lösungen für die Fälle FSI 1 und FSI 3 erzielt wurden. Bei dem Testfall FSI 2 hatten viele Löser Probleme mit der starken Netzdeformation. Besonders hervorzuheben ist die vergleichsweise geringe Rechenzeit, die der LB-Löser benötigte. Genaue Aussagen hierzu werden in dem Abschlussberichtsband der Forschergruppe veröffentlicht. Beim experimentellen Benchmark liegen zwar die Trajektorien der Endpunkte im Bereich derer des Experimentes, aber die gemessene Frequenz liegt deutlich über der experimentell ermittelten.

6 Dreidimensionale Testfälle

Ebenso wie der zweidimensionale Strömungslöser, wurde der dreidimensionale einer intensiven Validierung unterzogen. Im ersten Testfall werden die Druckbeiwerte auf einen Zylinder berechnet und mit Literaturergebnissen verglichen. Als erster Versuch der Fluid-Struktur-Interaktion wird eine sinkende Kugel in einem Rohr berechnet. Berechnungen mit der VIRTUALFLUIDS-ADHOC-Kopplung wurden anhand eines Benchmarks, der von Bathe [3] definiert wurde, validiert. Hier werden die Verschiebungen einer am Boden eingespannten, rechteckigen Platte berechnet. Abschließend wird eine längsangeströmte Winkelplatte simuliert, die den Abschluss der Beispiele bildet.

6.1 Druckbeiwerte eines Zylinders

Zur Validierung des dreidimensionalen Strömungslösers VIRTUALFLUIDS wurde die laminare Umströmung eines Zylinders untersucht. Dieser Benchmark ist eine Erweiterung des zweidimensionalen Benchmarks von Schäfer und Turek [103], der u. a. von Crouse [17] mit dem Lattice-Boltzmann-Verfahren auf nichtuniformen Gittern berechnet wurde. Die komplette Beschreibung kann man [103] entnehmen. Der Zylinder ist asymmetrisch in dem Strömungskanal positioniert. Es wird die stationäre Strömung von $Re = 20$ simuliert. Als Einflußrandbedingung ist ein Paraboloid mit der Funktion

$$u(0, y, z) = 16 u_{max} yz(H - y)(H - z)/H^4 \tag{6.1}$$

vorgegeben. Beim Ausfluß wird der Druck $p = 0$ gesetzt. Die Reynoldszahl ist durch

$$Re = \frac{\bar{u}D}{\nu} \quad mit \quad \bar{u} = \frac{4}{9} u_{max} \tag{6.2}$$

definiert. Die Kraftbeiwerte werden mit folgender Formel berechnet:

$$c_{D,L} = \frac{2F}{\rho \bar{u}^2 DH} \tag{6.3}$$

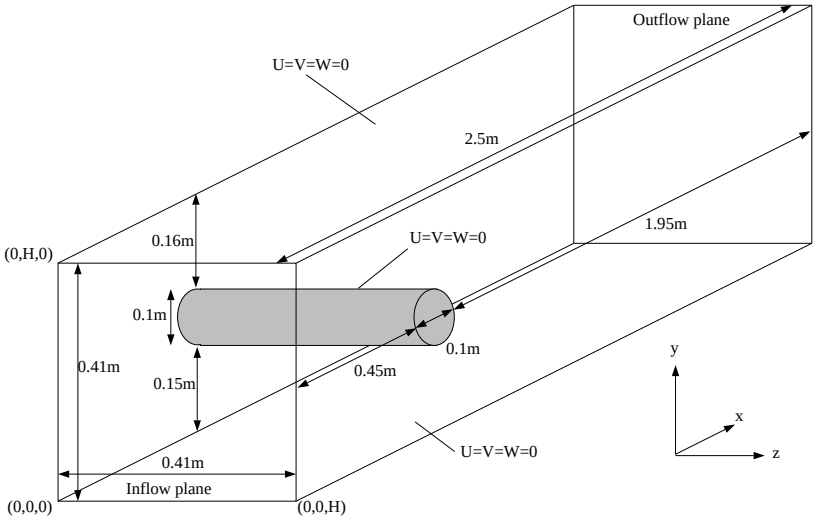


Abbildung 6.1: *Geometriedefinition und Randbedingungen für Strömung um einen Zylinder [103]*

Für die dreidimensionalen Fälle wurde die Fluid-Struktur-Interaktion im Rechenkern implementiert, der auf der Blockgitterdatenstruktur (siehe Kapitel 2.4) basiert. Es werden zwei verschiedene Gitterkonfigurationen untersucht. Im groben Gitter Konfiguration A gibt es $16 \times 4 \times 4$ Blöcke mit jeweils 7^3 Knoten pro Block. Dies entspricht einer Gebietsgröße von $112 \times 28 \times 28$ Knoten für die drei Raumrichtungen und somit insgesamt 87808 Knoten auf dem gröbsten Gitter. Das Gitter wird sukzessive um den Zylinder verfeinert. Das feine Gitter Konfiguration B hat genauso viele Blöcke wie das grobe, aber 11^3 Knoten pro Block. In Abbildung 6.2 sieht man ein a priori verfeinertes Blockgitter von Level 0 bis 3. Es sind nur die Blöcke dargestellt, die jeweils 7^3 bzw. 11^3 Knoten haben. Die Simulationsläufe, basierend auf nichtuniformen Gittern, zeigen quantitativ gute Ergebnisse, wie man Tabelle 6.1 entnehmen kann. Es ist hervorzuheben, dass die Multiskalensimulationen deutlich weniger Gitterknoten benötigen als die jeweiligen uniformen Simulationen. Das uniforme Gitter Level 3 hätte 44 Millionen Knoten. Das verfeinerte Gitter Level 0-3 hat hingegen 2 Millionen Knoten. Das entspricht einer Reduktion der Knotenzahl um den Faktor 22. Um gute Ergebnisse zu erzielen, genügt es, ein feines Gitter im Interessenbereich zu definieren.

Tabelle 6.1: *Drag und Lift auf einen Zylinder*

Konfiguration	Gitterlevel	c_D	c_L
A	0	7.011	-0.3
A	0-1	6.29	0.00565
A	0-2	6.139	0.015
A	0-3	6.1614	0.00813
B	0	6.2826	0.0110
B	0-1	6.0798	0.0104
B	0-2	6.0905	0.0076
Referenz		$6.05 - 6.25$	$0.008 - 0.01$

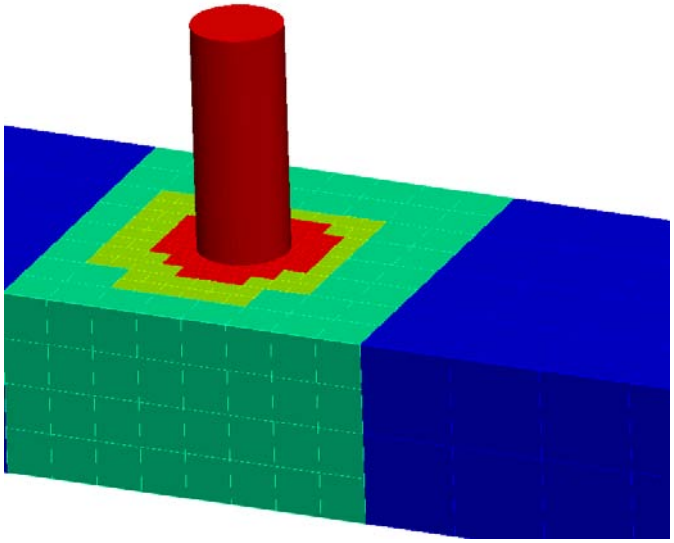


Abbildung 6.2: *a priori verfeinertes Blockgitter um den Zylinder*

6.2 Validierung einer Kugel

Als erster Fluid-Struktur-Interaktionstestfall in 3-D wird eine in einem Rohr sinkende Kugel simuliert. Zur Interaktion wurde der Fluidlöser VIRTUALFLUIDS mit der Starrkörperberechnungsmaschine PHYSICSENGINE (PE) [54] gekoppelt. Die Engine lehnt sich stark an bereits existierende Frameworks zur Simulation von Starrkörpern an. Im Gegensatz zu diesen Engines ist der primäre Fokus der PE die physikalisch exakte Simulation der Interaktion von Starrkörpern. Die Softwarestruktur von PE ermöglicht eine einfache Integration verschiedener Löser.

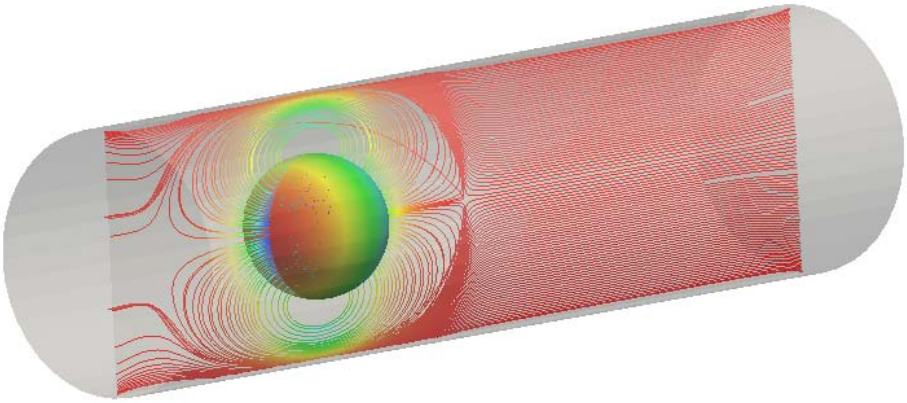


Abbildung 6.3: *Momentaufnahme einer umströmten Kugel in einem Kanal*

Der Benchmark ist folgendermaßen definiert: Eine Kugel mit dem Radius r und der Dichte ρ_s sinkt in einem Fluid der Dichte ρ_f und Viskosität ν.

Bei gleichförmiger Sinkgeschwindigkeit der Kugel stehen das Eigengewicht der Kugel F_G mit dem statischen Auftrieb F_A und der Widerstandskraft F_D im Gleichgewicht.

$$F_G = \frac{4}{3}\pi r^3 \rho_s g \tag{6.4}$$

$$F_A = \frac{4}{3}\pi r^3 \rho_f g \tag{6.5}$$

$$F_D = \frac{1}{2} C_{DW} \rho_f \pi r^2 U^2 \tag{6.6}$$

Somit ergibt sich für die Sinkgeschwindigkeit einer Kugel (im unendlich ausgedehnten Raum):

$$U = \sqrt{\frac{4}{3}\frac{(\rho_s - \rho_f)}{\rho_f}\frac{2r}{C_{DW}}g} \tag{6.7}$$

Da der erweiterte Kraftbeiwert C_{DW} abhängig von der Reynoldszahl $Re = Ud/\nu$ der Kugel ist, wird die Sinkgeschwindigkeit iterativ ermittelt.

Der Kraftbeiwert C_D berechnet sich für laminare Strömungen im Stokesregime mit

$$C_D = 24/Re \tag{6.8}$$

Schiller und Naumann [106] erweiterten diese Gleichung für höhere Reynoldszahlen.

$$C_D = \frac{24}{Re}(1 + 0.15Re^{0.687}) \tag{6.9}$$

Für eine endliche Weite des Rohres, in dem die Kugel sinkt, müssen weitere Korrekturen berücksichtigt werden. Fayon und Happel [25] leiteten dies als Störungserweiterung von $\lambda = r_{Kugel}/r_{Zylinder}$ her:

$$C_{DW} = C_D + \frac{24}{Re}(K - 1) \tag{6.10}$$

mit

$$K = \frac{1 - 0.75857\lambda^5}{1 - 2.105\lambda + 2.0865\lambda^3 - 1.7068\lambda^5 + 0.72603\lambda^6} \tag{6.11}$$

Gleichung 6.10 gilt für $\lambda < 0.6$ und $Re < 50$ und hat einen relativen Fehler von $\pm 5\%$.

Für den laminaren Bereich von $Re < 0.5$ gilt das Stokessche Reibungsgesetz. Hier ist die Widerstandskraft F_D gegeben durch

$$F_D = 6\pi r \nu \rho_f U \tag{6.12}$$

Damit ergibt sich für diesen Bereich eine Sinkgeschwindigkeit von

$$U = \frac{2}{9}\frac{(\rho_s - \rho_f)}{\rho_f}\frac{gr^2}{\nu} \tag{6.13}$$

Diese Formel gilt für eine unbegrenzt ausgedehnte Flüssigkeit. D.h. die Wände des Zylinders sind in dieser Formel nicht berücksichtigt.

6.2.1 Ortsfeste Kugel

Der Fluidbereich ist durch ein zylindrisches Rohr definiert. Eine ortsfeste Kugel mit Haftrandbedingungen ist in der Mittelachse des Rohres positioniert. Die Kugel ist ein Drittel der Kanallänge vom Einfluss entfernt. Am Einfluß und an den Kanalwänden wurden Geschwindigkeitsrandbedingungen gesetzt, um eine bewegte Kugel zu simulieren. Der Ausfluß hat einen Druckrand ($p = 0$). In Abbildung 6.4 ist das Gebiet schematisch dargestellt. Die Anströmkraft wurde mit der Impulsaustauschmethode berechnet. In Tabelle 6.2 sind die Ergebnisse dargestellt. Die Simulation wurde für eine Reynoldszahl von $Re = 1$ und drei verschiedene Auflösungen durchgeführt. Für die Kollision wurde das Two-Relaxation(TRT)-Modell verwendet. Die Dichte des Fluids beträgt $1000\frac{kg}{m^3}$. Der Referenzwert für den Dragbeiwert ist $C_{DW} = 144.48$. Die Rechnung wurde beendet, wenn die Änderung des Dragbeiwertes kleiner als 10^{-6} war.

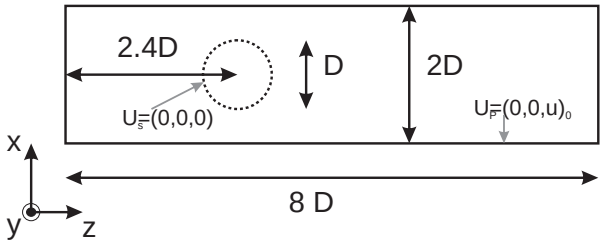


Abbildung 6.4: *Setup der numerischen Simulation*

Tabelle 6.2: *Kraftbeiwerte auf Kugel für verschiedene Netzgrößen für* $Re = 1$

Gebietsgröße	u_0 $[m\,s^{-1}]$	ν $[m^2\,s^{-1}]$	$D/\Delta x$	C_{DW} [-]	rel. Fehler [%]
$30^2 \times 120$	0.0111	0.166	15	141.19	2.3
$60^2 \times 240$	0.0055	0.166	30	142.00	1.7
$90^2 \times 360$	0.0037	0.166	45	142.30	1.5

6.2.2 Gleichförmig bewegte Kugel

Nachdem gezeigt wurde, dass die Kraftbeiwerte für eine ortsfeste Kugel bis auf ein Prozent Genauigkeit berechnet werden können, wird dasselbe Gebiet mit einer gleichförmig bewegten Kugel simuliert und der resultierende Kraftbeiwert gemessen. Die Zylinderwände sowie der Einfluß und Ausfluß werden mit Haftrandbedingungen (no-slip) simuliert. Die Kugel wird im Abstand von einem Kugeldurchmesser zum Einflussrand ins Fluid gesetzt und entlang der Zylindermittelachse gleichförmig bewegt. Ein konvergenter Kraftwert stellt sich im groben Setup nach 500 Iterationen und im feinen Setup nach 4000 Iterationen ein. In Tabelle 6.3 sind die Ergebnisse dargestellt.

Tabelle 6.3: *Kraftbeiwerte auf gleichförmig bewegte Kugel für verschiedene Netzgrößen für* $Re = 1$

Gebietsgröße	u_0 $[m\,s^{-1}]$	ν $[m^2\,s^{-1}]$	$D/\Delta x$	C_{DW} [-]	rel. Fehler [%]
$30^2 \times 120$	0.0111	0.166	15	139.53	3.4
$60^2 \times 240$	0.0055	0.166	30	141.64	1.9
$90^2 \times 360$	0.0037	0.166	45	142.38	1.4

6.2.3 Sinkende Kugel

Die in einem Fluid sinkende Kugel wurde mit verschiedenen Auflösungen sowie verschiedenen Kugeldichten simuliert. Die Simulationsparameter für den Testfall sind: $r_{Zylinder} = 0.1m$, $r_{Kugel} = 0.05m$, $\nu = 0.01m^2/s$ und $\rho_f = 1000kg/m^3$. Die Zylinderlänge war $L = 0.8m$, so dass ein stationärer Zustand erreicht wird, bevor die Kugel auf den Boden trifft. Die Ränder haben eine Haftrandbedingung (no-slip). Auf das Fluid wird eine Volumenkraft entsprechend Gleichung 4.18 zur Umrechnung der Gewichtskraft aufgegeben. Bei der Kugel wird die Geschwindigkeit entsprechend der modifizierten Bounce-Back-Randbedingung zweiter Ordnung auf das Fluid übertragen. Aufgrund der zeitintensiven Berechnungen sowie des enorm hohen Speicherbedarfs der hochaufgelösten Konfigurationen wurden diese mit der parallelen Simulationskernerweiterung von VIRTUALFLUIDS simuliert. Dieser wird in der Dissertation von Freudiger [32] beschrieben. Mit diesem Testfall wird gezeigt, dass die verteilt rechnende Version von VIRTUALFLUIDS auch in der Lage ist mit beweglichen Geometrien, wie sie in der Fluid-Struktur-Interaktion auftreten, umzugehen. Die Berechnung mit der höchsten Auflösung hatte 37 Millionen Knoten.

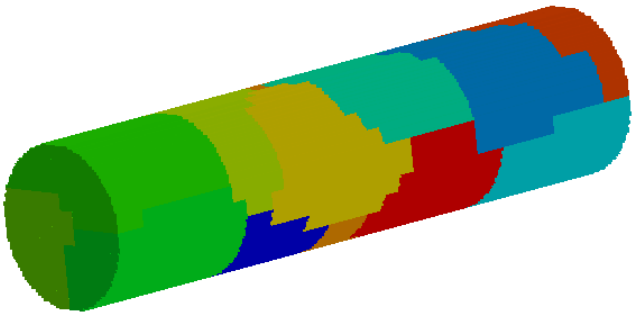


Abbildung 6.5: *Partitionierung des Strömungsfeldes bei einer verteilten Berechnung mit 14 Subdomains*

Für die Netz- bzw. Machzahlstudie hat die Kugel eine doppelt so hohe Dichte $\rho_s = 2000 kg/m^3$ wie das Fluid. Die iterativ ermittelte analytische Geschwindigkeit beträgt $u_{ref} = 0.0906$. Das ergibt eine Reynoldszahl von $Re = 0.91$. Die resultierenden Kräfte der Simulation schwanken um ca. 0.1% um die Größe der Gewichtskraft der Kugel. In Tabelle 6.4 sind die gemessenen Geschwindigkeiten für drei verschiedene Auflösungen dargestellt. Die Viskosität im LB-Gitter wurde mit 0.1666 festgesetzt. In Abbildung 6.6 ist eine Zeitserie für die Geschwindigkeit der fallenden Kugel dargestellt.

Tabelle 6.4: *Geschwindigkeiten der fallenden Kugel für verschiedene Netzgrößen*

Gebietsgröße	$D/\Delta x$	$u\ [m\,s^{-1}]$	rel. Fehler [%]
$30^2 \times 120$	15	0.09080	0.2
$60^2 \times 240$	30	0.09215	1.7
$90^2 \times 360$	45	0.09186	1.4
$120^2 \times 480$	60	0.09190	1.4
$150^2 \times 600$	75	0.09180	1.3
$180^2 \times 720$	90	0.09177	1.3
$210^2 \times 840$	105	0.09174	1.2

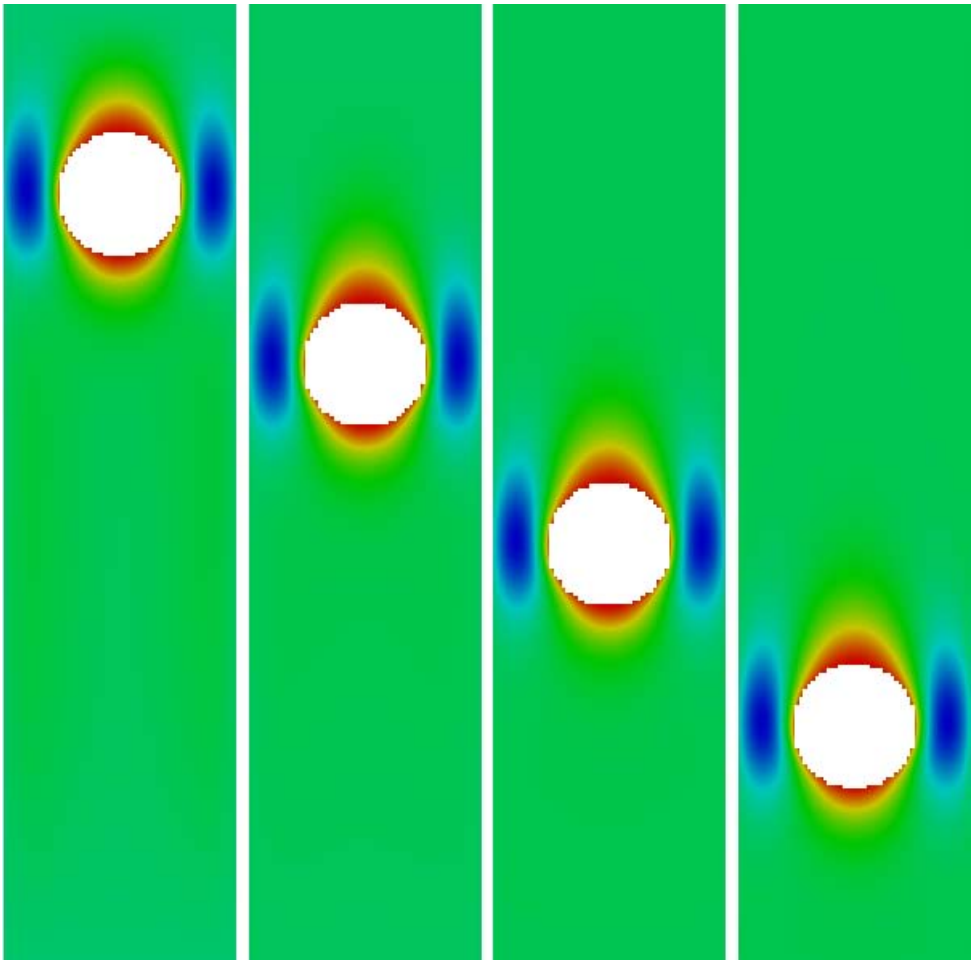


Abbildung 6.6: *Zeitserie der Geschwindigkeit in Sinkrichtung*

Tabelle 6.5 zeigt den Fehler der Geschwindigkeiten als Funktion der Strukturdichten. Die Auflösung des Gebietes betrug $30^2 \times 120$ Knoten. Je größer die Masse der Kugel, desto höher ist die Sinkgeschwindigkeit. Die ermittelten Fehler befinden sich innerhalb der Modelltoleranzen von 5% (Gleichung 6.10).

Tabelle 6.5: *Geschwindigkeiten der fallenden Kugel für verschiedene Strukturdichten*

ρ_s	Re	$u_{analytisch}[m\,s^{-1}]$	$u_{gemessen}[m\,s^{-1}]$	rel. Fehler [%]
1200	0.18	0.0184	0.0187	1.6
1500	0.46	0.0457	0.0468	2.4
2000	0.91	0.0906	0.0896	1.1
4000	2.65	0.2653	0.2740	3.3
6000	4.34	0.4340	0.4500	3.7
8000	5.98	0.5977	0.6250	4.5
10000	7.58	0.7578	0.7800	2.9

6.3 Längsangeströmte Rechteckplatte

Ein weiterer Testfall für die Fluid-Struktur-Interaktion ist eine längsangeströmte Rechteckplatte. Bathe und Ledezma schlagen diesen als FSI-Benchmark in ihrer Veröffentlichung [3] vor. Bei diesem Fall wird die Auslenkung der Platte mit verschiedenen Reynoldszahlen in einem Spektrum von $Re = 500$ bis $Re = 5000$ ermittelt. Das Fluid wird mit der Arbitrary-Lagrangian-Eulerian (ALE) Formulierung beschrieben, die mit einer Lagrange-Formulierung der Struktur gekoppelt ist. Die Berechnung resultiert in einem quasi-stationären Zustand der Platte.

Bei der Berechnung mit dem Lattice-Boltzmann-Löser VIRTUALFLUIDS, gekoppelt mit dem p-FEM-Löser ADHOC, werden die zwei Fälle $Re = 500$ und $Re = 2500$ untersucht. Zusätzlich wird ein laminarer Testfall bei $Re = 10$ untersucht.

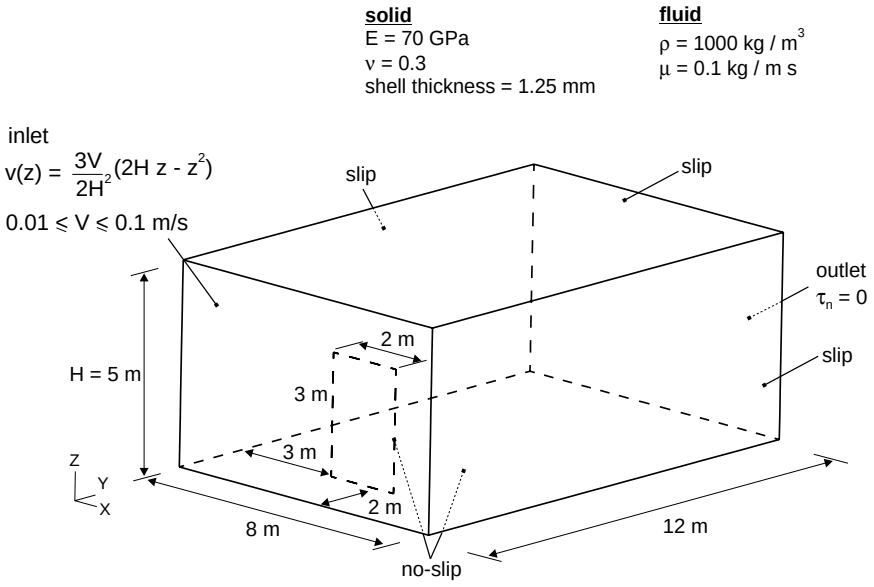


Abbildung 6.7: *Berechnungsgebiet mit integrierter Rechteckplatte [3]*

Die Testfallkonfiguration (vgl. Abbildung 6.7) wurde folgendermaßen definiert: Die angeströmte Platte besitzt ein E-Modul von $E = 7 \times 10^{10} \frac{N}{m^2}$, eine Querdehnzahl von $\nu_s = 0.3$ und eine Plattendicke von $h = 0.00125m$. Das Fluid wird in einer quaderförmigen Box simuliert. Der Boden sowie die angeströmte Platte haben Haftrandbedingungen (no-slip), die Seitenwände und Decke Rutschrandbedingungen (slip), der Einlass ein Geschwindigkeitsprofil und der Auslass einen konstanten Druck von $p = 0N/m^2$. Die Dichte des Fluids beträgt $\rho = 1000 \frac{kg}{m^3}$ und die dynamische Viskosität $\mu = 0.1 \frac{kg}{ms}$. Die kinematische Viskosität ist demzufolge $\nu = 0.0001 \frac{m^2}{s}$. Die Geschwindigkeit am Einströmrand ist in Abhängigkeit von der Kanalhöhe H definiert:

$$v(z) = \frac{3V}{2H^2}(2Hz - z^2) \tag{6.14}$$

Die Reynoldzahl bezieht sich auf die Kanalhöhe von $5m$. Die Platte wurde mit einer Auflösung von $10 \times 1 \times 18$ Elementen und einem Ansatzgrad von drei diskretisiert. Das Seitenverhältnis der Elemente ist $1 : 160$. Das Fluidgebiet wurde mit einem uniformen Netz sowie lokaler Verfeinerung berechnet. Die Machzahl beträgt $Ma = 0.05$.

6.3.1 Abschätzformel

Für die verschiedenen Anströmgeschwindigkeiten u wurde die Auslenkung w mit der eindimensionalen Kragarmformel [107] berechnet:

$$\mathrm{w} = \frac{1}{8}\frac{ql^4}{EI} \tag{6.15}$$

Die Steifigkeit EI beträgt

$$EI = \frac{Eh^3}{12(1-\nu_s^2)} \tag{6.16}$$

und die Belastung q ist näherungsweise

$$q = \frac{F}{A} = \frac{1}{2}c_d\rho u^2. \tag{6.17}$$

6.3.2 Testfall Re=10

In diesem Testfall wird für verschiedene Anströmgeschwindigkeiten die jeweilige Auslenkung bestimmt. Die Konvergenz der Auslenkung wird in Abbildung 6.8 gezeigt. Es wird deutlich, dass das System nach wenigen Schwingperioden einen stationären Zustand erreicht. Tabelle 6.6 zeigt die Studie für verschiedene Anströmgeschwindigkeiten für zwei Gitterauflösungen. Es ist ein konsistentes Verhalten zu erkennen und die Ergebnisse liegen im Bereich der Abschätzung mit der Kragarmformel 6.15. Um systembedingte Fehler auszuschließen, wurde der aerodynamische Kraftbeiwert von $c_d = 9.4$ mit ANSYS CFX bestimmt.

Tabelle 6.6: *Auslenkung der längsangeströmten Platte für* $Re = 10$

Gebietsgröße	$u\ [m/s]$	$\nu\ [m^2/s]$	Auslenkung Kragarm$[m]$	Auslenkung $[m]$
$96 \times 144 \times 60$	0.0002	0.0001	1.52e-4	1.656e-4
$96 \times 144 \times 60$	0.002	0.001	1.52e-2	1.645e-2
$96 \times 144 \times 60$	0.02	0.01	1.52	1.235
$192 \times 288 \times 120$	0.0002	0.0001	1.52e-4	1.690e-4
$192 \times 288 \times 120$	0.002	0.001	1.52e-2	1.683e-2
$192 \times 288 \times 120$	0.02	0.01	1.52	1.249

*

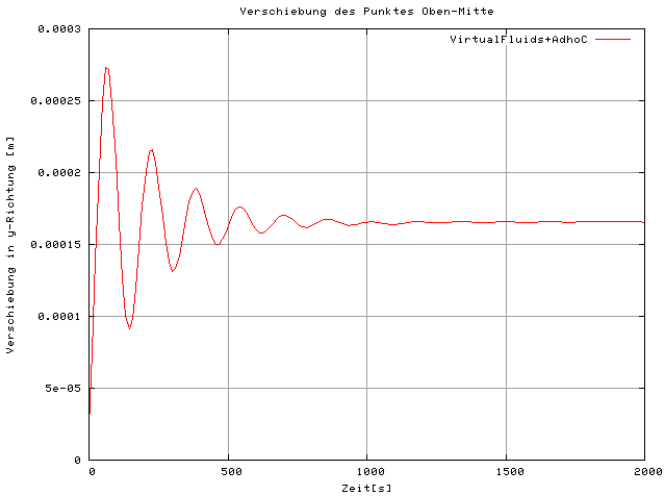


Abbildung 6.8: *Auslenkung der Platte für* $Re = 10$

6.3.3 Testfälle Re=500 und Re=2500

Für höhere Reynoldszahlen ist die Auslenkung der Platte überlagert von Oszillationen mit einer Amplitude von max. $10^{-4}m$, die durch ablösende Wirbel der zeitabhängigen Strömung erzeugt werden. Die Wirbel lösen sich an der Plattenoberkante ab und advektieren durch das Fluidfeld. Das Gebiet wurde mit einer Auflösung von $96 \times 144 \times 60$ Knoten diskretisiert und bis zu zwei Gitterlevel im Bereich der ausgelenkten Platte verfeinert (Abbildung 6.9). In Abbildung 6.10 und 6.12 ist der zeitliche Verlauf der Verschiebungen für die zwei Testfälle gezeigt. Abbildung 6.11 und 6.13 zeigen Momentaufnahmen der Stromlinien im Fluidquerschnitt, bei der die komplexe Struktur der Strömung deutlich wird.

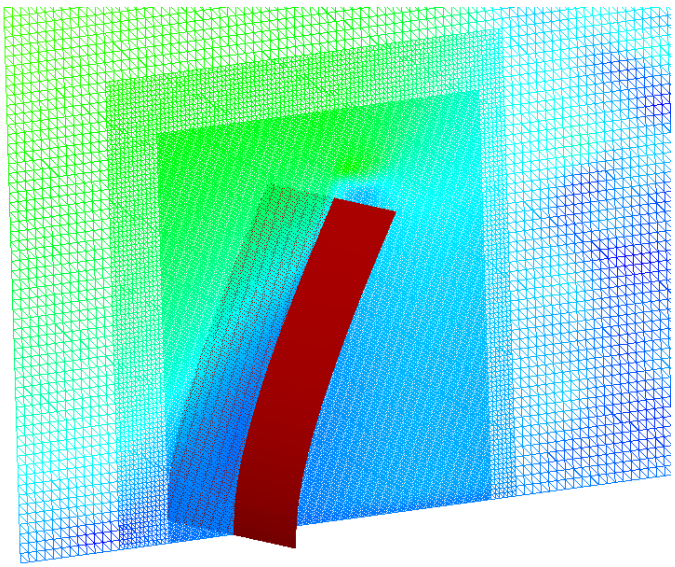


Abbildung 6.9: *Gitterverfeinerung im Plattenbereich*

Für den Testfall $Re = 500$ beträgt der berechnete Wert für die Auslenkung $0.0897m$ auf dem uniformen

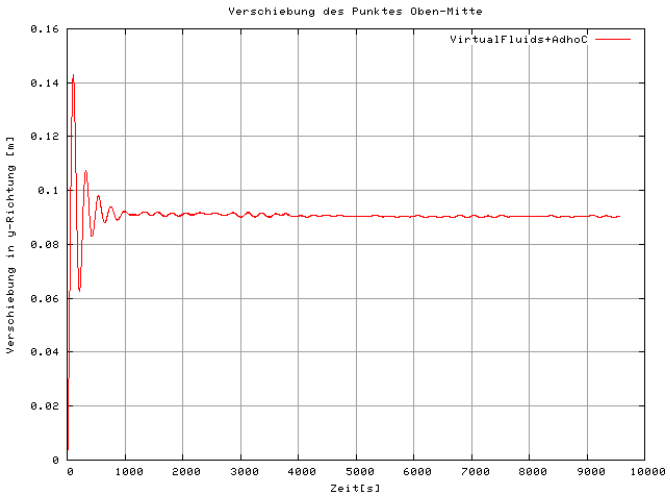


Abbildung 6.10: *Auslenkung der Platte für* $Re = 500$

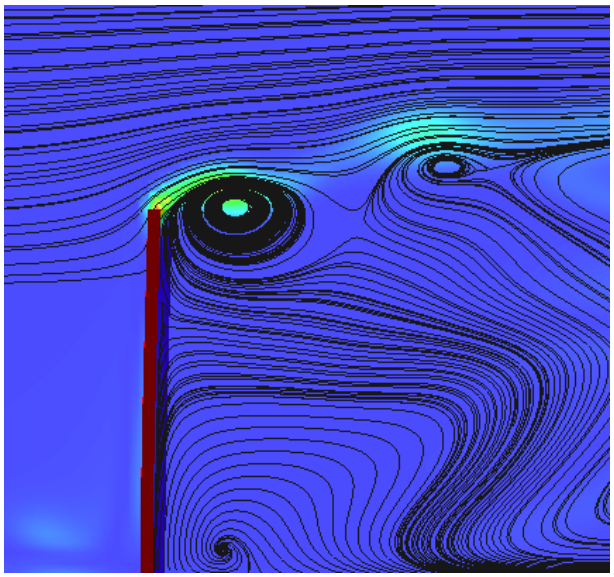


Abbildung 6.11: *Momentaufnahme der längsangeströmten Rechteckplatte bei* $Re = 500$ *(Stromlinien, Wirbelstärke)*

Berechnungsgebiet. Die Fluktuationen (vgl. Abbildung 6.10) schwanken um vier Promille bezogen auf die Auslenkung. Mit einer Verfeinerung von einer Stufe bzw. zwei Stufen beträgt der Wert $0.0891m$ bzw. $0.0893m$. Bathe und Ledezma erhalten einen stationären Wert von $0.066m$. Bei $Re = 2500$ ist die Auslenkung $1.351m$. Der Wert beträgt $1.349m$ bei einer Verfeinerung von einem Level und $1.352m$ mit einer Verfeinerung von zwei Leveln. Bathe und Ledezma hingegen haben einen Wert von $0.98m$ berechnet. Die Abweichungen der beiden Testfälle beträgt über 30 Prozent.

Zusammenfassend ergibt die Simulation qualitativ zufriedenstellende Ergebnisse. Da weitere Referenzergebnisse von anderen FSI-Codes fehlen, ist eine Bewertung der Rechenwerte unklar. Bathe und Ledezma haben die Platte mit einer relativ groben Diskretisierung berechnet. So hatte das feinste Fluidnetz ca. 46.000 Elemente. Die höchste Auflösung mit der LB-Methode hatte $6.762.588$ Knoten. Erste Untersuchungen mit dem Fluid-Struktur-Löser ANSYS MULTIPHYSICS für den Testfall $Re = 500$ zeigen eine Tendenz in Richtung der Lösung von VIRTUALFLUIDS-ADHOC. Die Verschiebung beträgt $0.094m$ mit ca 570.000 Elementen. Die Berechnung dauert ca. 4 Tage auf 4 Prozessoren. Die uniforme LB-Berechnung hingegen dauerte ca. einen Tag auf einem Prozessor. Eine genauere Untersuchung der Ergebnisse einschließlich Konvergenzstudien und Vergleichsrechnung mit ANSYS MULTIPHYSICS ist Gegenstand weiterer Arbeiten.

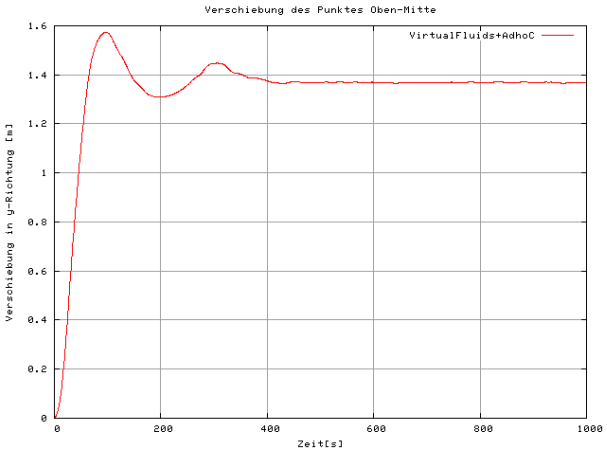


Abbildung 6.12: *Auslenkung der Platte für* $Re = 2500$

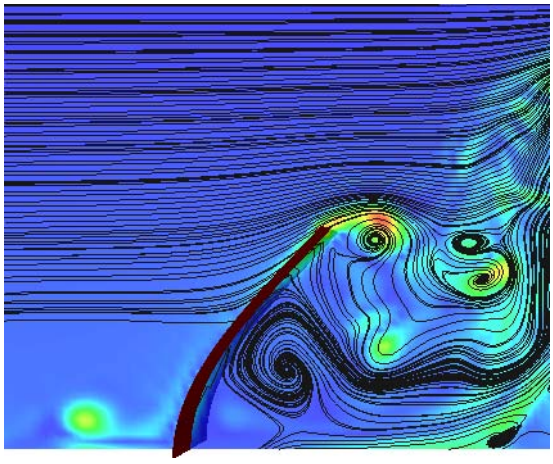


Abbildung 6.13: *Momentaufnahme der längsangeströmten Rechteckplatte bei $Re = 2500$ (Stromlinien, Wirbelstärke)*

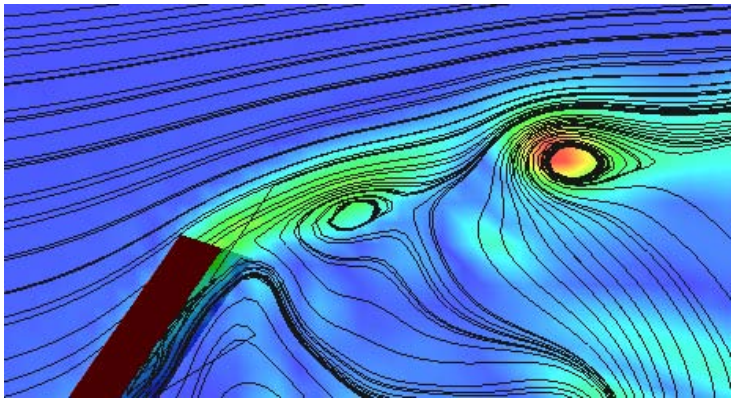


Abbildung 6.14: *Detail der Momentaufnahme der längsangeströmten Platte bei $Re = 2500$ (Stromlinien, Wirbelstärke)*

6.4 Zusammenfassung

Der 3-D-Löser VIRTUALFLUIDS wurde für FSI validiert. Der Vergleich der Druckbeiwerte auf einen Zylinder mit dem DFG-Benchmarkergebnissen zeigt zufriedenstellende Ergebnisse. Bei der Simlation der sinkenden Kugel betrug die Abweichung der resultierenden Geschwindigkeit nur ein Prozent von der semianalytischen Lösung (Gleichung 6.7). Als erster Fluid-Struktur-Interaktionstestfall mit dem Finite Element Strukturlöser ADHOC wurde die Simulation einer längsangeströmten Platte durchgeführt. Für den stationären sowie den transienten Testfall ergaben sich Ergebnisse, die mit der Kragarmformel abgeschätzt bzw. von Bathe und Ledezma errechnet wurden. Die Validierungsrechnungen sind jedoch nur ein erster Schritt. Es müssen Konvergenzstudien durchgeführt und die Ergebnisse mit anderen Codes bzw. Experimentem gegengeprüft werden. Gegenstand weiterer Forschungen sollte auch die Simulation mit Reynoldszahlen im turbulenten Regime sein. Für das LB-Verfahren vorhandene Turbulenzmodelle müssen hier im Zusammenhang mit der Fluid-Struktur-Kopplung validert werden.

7 Prototyp einer interaktiven Simulationsumgebung

Eine effiziente Bearbeitung, Auswertung und Optimierung physikalischer Problemstellungen ist nur möglich, wenn die Lösungsumgebung u. a. in der Lage ist, Adaptivität, Parallelisierung und wissenschaftliche Visualisierung zu vereinen. Um dieses Ziel zu erreichen, wird der klassische Ansatz - Präprozess, Berechnung, Postprozess - ersetzt durch einen Computional-Steering-Ansatz. Dies führt zu einer interaktiven Umgebung, in der der Benutzer sowohl Randbedingungen und Geometrien als auch Berechnungsparameter und die Darstellung zur Simulationslaufzeit ändern kann. Im Rahmen dieser Arbeit wurde das erweiterbare Framework VIRTUALFLUIDS entwickelt, das auf objekt-orientierten Softwaretechnologien basiert. Es wird die Integration verschiedener numerischer Rechenkerne (z. B. für freie Oberflächen, Flachwasser, Fluid-Struktur-Interaktion) sowie eine Kopplung an eine Visualisierung gezeigt.

Ein Großteil der Umsetzung basiert auf der vorhandenen C++-Softwarebibliothek, die im Laufe der vergangenen Jahre am Institut für rechnergstütze Modellierung im Bauingenieurwesen in Zusammenarbeit mit Sören Freudiger erstellt und stetig weiterentwickelt wurde. Die Erkenntnisse mehrerer Studien- und Diplomarbeiten [56, 63, 81, 89, 104] dienten hierbei der stetigen Weiterentwicklung der Bibliothek und somit auch zur Entwicklung der Simulationsumgebung.

Grundlegendes Ziel der Entwicklungsarbeit war es, einen adaptiven, parallelen, interaktiven Lattice-Boltzmann-Multi-Physik-Strömungslöser zu erstellen. Geometrische und numerische Modellierung kann in einer Virtual-Reality-Umgebung stattfinden und geometrische, physikalische sowie visuelle Parameter können während des Rechenprozesses interaktiv manipuliert werden. Weiteres Ziel ist es, eine hohe Rechenleistung mit den Grundprinzipien der Softwareentwicklung zu vereinen.

Geometrische Hindernisobjekte sowie Verfeinerungsregionen, Container für die Mengen von den jeweiligen Objekten, topologische Rechengitter, Änderungskriterien, Randbedingungen für die Rechengitter und Dateiein- sowie ausgaberoutinen sind die strukturelle Basis des Strömungslösers. Eine abstrakte Reduzierung der hier genannten Bestandteile hat den Vorteil, das weitere Mitarbeiter sich nicht mehr um zeitraubende Routinen im Sinne von Implementierung und Validierung kümmern müssen und eigene notwendige Komponenten durch Verwendung der Vorhandenen erstellt werden können.

Das hier vorgestellte Softwarekonzept hat seinen Ursprung im *Quadtree based Toolkit for Finite Volume Modelling for the Shallow Water Equation*, das am Lehrstuhl für Bauinformatik in Cottbus entwickelt wurde [9, 36]. Weiterhin wurde diese Arbeit durch institutseigene Forschungen für einen virtuellen Windkanal inspiriert.

Unabhängig vom Strömungslöser sind in dem Softwarepaket des Institutes viele nützliche Bibliotheken, die alternativ genutzt werden können. Beispiele sind die Geometriebibliothek und ein Funktionsdiagramm.

Traditionell werden numerische Löser von Experten für spezifische Probleme entwickelt. Sie sind somit meist eingeschränkt, was die Anpassungsfähigkeit, Erweiterbarkeit, Wiederverwendbarkeit und Nachvollziehbarkeit betrifft.

Bei der Modellierung in der Bauinformatik sind Transfer und Generalisierung Stufen der Softwareentwicklung. Transfer bezeichnet die Übertragung bestehender Softwarekomponenten auf neue Aufgabenstellungen. Generalisierung ist die Verallgemeinerung von bereits vorhandenen Softwaremodulen, um sie auch auf andere Problemstellungen anwendbar zu machen. Für viele numerische Problemstellungen wird die Methode des Transfers genutzt. Man kopiert seinen ”alten” Code und implementiert z. B. neue Terme. Auf eine Generalisierung, oft mit dem falschen Vorwand der Ineffizienz, wird weitestgehend verzichtet.

Die Gesamteffizienz wird beeinflusst durch die Ausführgeschwindigkeit des numerischen Rechenkerns als auch die Zeit für dessen Entwicklung [11]. Bei dieser Arbeit wurde vielfach die Generalisierung in den Vordergrund gestellt, um die Gesamteffizienz, also die Übertragung auf andere Problemstellungen, zu verbessern.

Das nachfolgende Kapitel zeigt den Entwicklungsstand der Softwareplattform, die die gleichzeitige Berechnung, Beeinflussung und Betrachtung von Simulationsdaten ermöglicht. Grundlage der Berechnung bilden Modelle basierend auf der Lattice-Boltzmann-Methode zur Simulation von Strömungen mit freier Oberfläche, Einphasen- und Mehrphasenströmungen.

Im Rahmen dieser Arbeit wurde der Prototyp eines immersiven Werkzeugs entwickelt, um dem Ingenieur das Modellieren innerhalb einer Virtual-Reality-Umgebung zu ermöglichen. Diese bietet dem Betrachter neben einer räumlichen Darstellung mit echter Tiefeninformation zusätzlich die Möglichkeit, sich innerhalb der Szene zu bewegen und Objekte zu manipulieren. Dem Ingenieur wird so ein intuitives Arbeiten ermöglicht.

In der Bibliothek existieren zwei- sowie dreidimensionale Berechnungskerne. Es wird außerdem unterschieden zwischen dem knotenbasierten und dem blockstrukturierten Datenstrukturen. Diese vier Zweige sind vom Grundaufbau ähnlich. Deswegen wird hier lediglich die Softwarestruktur des zweidimensionalen knotenbasierten Codes beschrieben.

Das Design richtet sich nach den grundlegenden Softwareentwicklungsprinzipien, die u. a. in [35] ausführlich beschrieben sind. Zunächst wird das grundlegende Softwaremodell für die Topologie des Rechengitters, die Gittergenerierung und spezifische Gitter erläutert. Daraufhin wird die Visualisierungspipeline beschrieben und die grafische Nutzerschnittstelle vorgestellt. Abschließend wird auf die Parallisierung der Datenstruktur und die aspektorientierte Programmierung im Zusammenhang mit der Entwicklung des Frameworks eingegangen.

7.1 Das generalisierte Quadtree-Modell

Die Basis, auf der das Softwarekonzept aufbaut, wird in der Habilitationschrift von Brüggemann [11] unter dem Kapitel *Quadtree-basierter Baukasten für die Simulationsmodellierung* ausführlich beschrieben. Es existieren disziplin- und anwendungsunabhängige Klassen und Schnittstellen als Basis für eine disziplinspezifische und anwendungsabhängige Spezifikation. Zwei Hauptklassen, das *NodeGrid* und der *Node*, stellen notwendige Informationen zur Verfügung, um baumarige Gitter zu erzeugen, diese zu editieren und in ihnen zu navigieren.

Das *NodeGrid* verwaltet die Menge der Knoten (*Nodes*) und beschreibt ein hierarchisches, baumstrukturiertes Rechengitter. Die Knoten kennen ihre Position x,y in Matrixkoordinaten und ihren Verfeinerungslevel. Das NodeGrid verwaltet die Knoten in einer Hashtabelle je Gitterlevel. Der Speicherzugriff erfolgt über einen Schlüssel (key), der sich aus den Matrixkoordinaten ergibt. Alternative Speichermöglichkeiten wären uniforme zwei- bzw. dreidimensionale Datenfelder für jeden Level bzw. eine Baumdatenstruktur (Quadtree). Teilbesetzte Datenfelder müssen vollständig initialisiert werden und benötigen sehr viel Speicher. Quadtrees sind zwar sehr effizient im Aufbau, aber gerade im Hinblick auf adaptive Berechnungen zu komplex, wenn man die zugehörigen Eltern-Kind-Beziehungen jeweils verzeigert.

Die Nachbarschaftssuche geschieht folgendermassen: Jeder Knoten fragt die Hashtabelle, ob er in dem Abstand seiner Hierarchiestufe einen Nachbarn hat. Der Nachbar eines Knotens ist immer in dem Knotenabstand des jeweiligen Knotenlevels. Hat der Knoten auf demselben Level keinen Nachbarn, so wird über Bitshiftoperationen die Position im darunterliegenden oder darüberliegenden Level bestimmt und in der Hashtabelle geprüft, ob der Knoten vorhanden ist.

Die *NodeGrid*- und *Node*-Klasse bilden die topologische Basis und stellen die Funktionalität zur Verfeinerung bzw. Vergröberung sowie zur Navigation bereit.

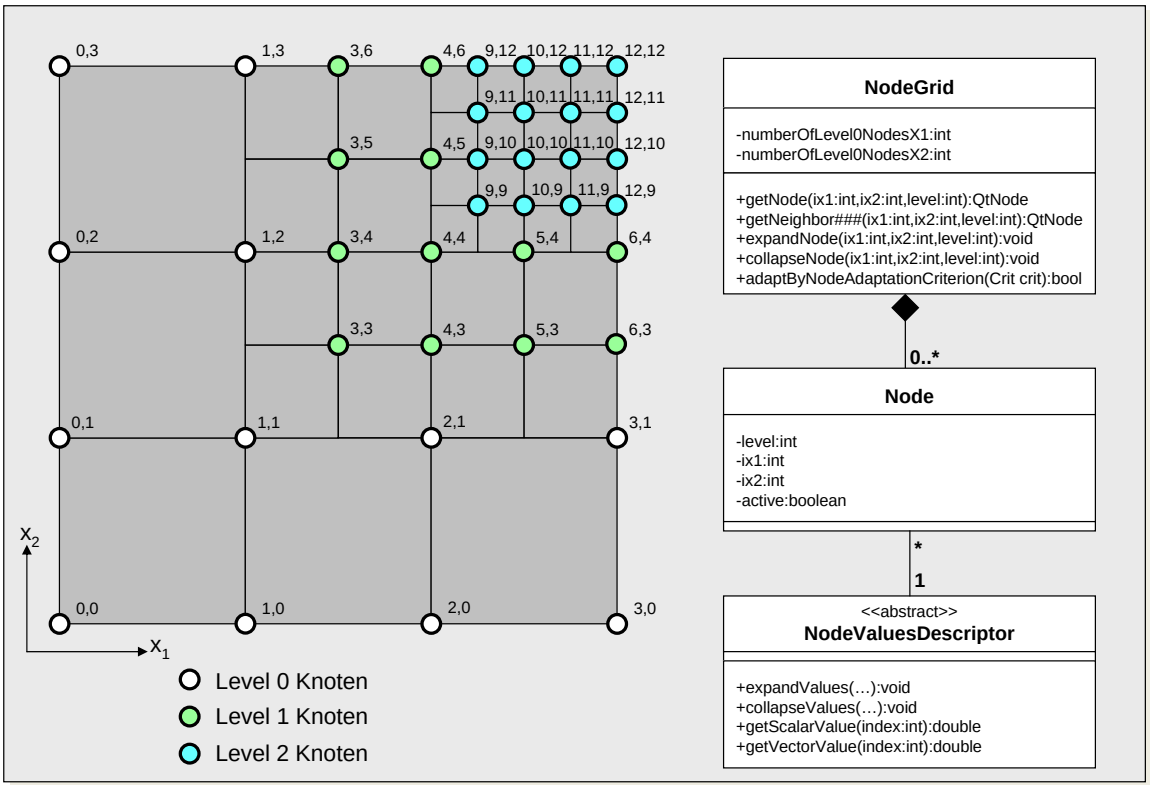


Abbildung 7.1: *Knotengitter: Topologie*

Die Beschreibung von spezifischen Werten einzelner Knoten erfolgt über die abstrakte *NodeValueDescriptor*-Klasse (NVD). In dieser sind Methoden wie *getScalarValue(Index)* bzw. *getVectorValue(Index)* für den allgemeinen Zugriff auf seine spezifischen Werte via Indices deklariert. Für die Verfeinerung sowie Vergröberung gibt es abstrakte Methoden, die in den spezifischen Klassen implementiert werden müssen.

Zur Beschreibung anwendungspezifischen Verhaltens existieren spezifische Adaptionskriterien, welche von der *NodeAdaptationCriterion*-Klasse abgeleitet sind.

In der *adaptByNodeAdaptationCriterion()*-Methode im *NodeGrid* wird für jeden Knoten die *adaptNode()*-Methode des spezifischen *NodeAdaptationCriterion* aufgerufen. Hier wird für jeden Knoten in einem kurzen Modul das spezifische Verhalten implementiert. Dies sind zum Beispiel Verfeinerungs- sowie Vergröberungskriterien oder auch Berechnungskriterien. In Abbildung 7.2 sind einige implementierte Kriterien dargestellt. Der *SwSurfaceElevationGradientAdapter* verfeinert bzw. vergröbert beispielsweise bei einer Flachwassersimulation das Gitter abhängig von spezifischen Schwellwerten des Oberflächengradienten der Wellenfront. Der *VorticityAdapter* verfeinert das Berechnungsgitter abhängig von der Wirbelstärke. Ein geometrisches Kriterium ist der *InsideGeoObject2DRefineAdapter*. Hier wird das Gebiet innerhalb einer gegebenen Geometrie verfeinert. Schließlich dient der *NodeRatioAdapter* zum Glätten des Gitters, sodass nach der Verfeinerung ein Verhältnis von 2:1 für den Knotenabstand an den Gitterübergangsgrenzen vorliegt.

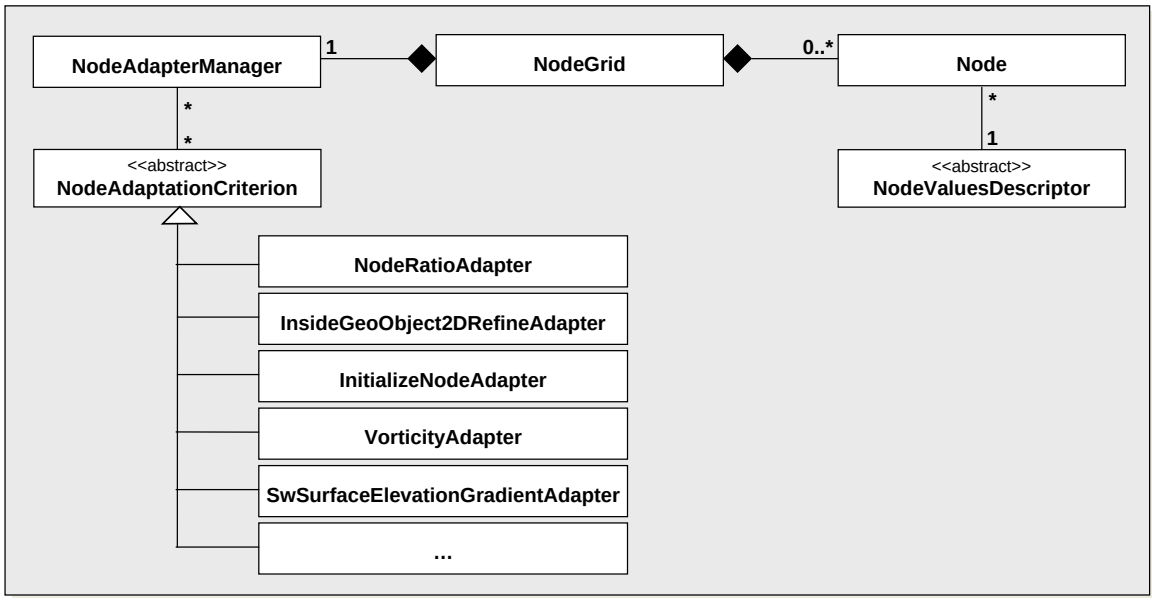


Abbildung 7.2: *UML-Diagramm spezifischer Adaptionskriterien*

7.2 Geometrie / Gittergenerierung

Das *NodeGrid* ist ein Container für Knoten (*Nodes*), die in Matrixkoordinaten abgespeichert sind. Die geometrischen Objekte (*GeoObject2D*) hingegen besitzen reale Weltkoordinaten. Geometrische Objekte können in 2-D z. B. ein Kreis, ein Rechteck, eine Linie und ein Polygon sein. Das zu untersuchende Gebiet ist durch Matrixkoordinaten vernetzt, die in reale Weltkoordinaten transformiert werden. Dies erfolgt durch eine einheitliche Koordinatentransformation, die in der Klasse *CoordinateTransformation2D* implementiert ist. Bei der Gittergenerierung hat man entweder die Möglichkeit die Gitterknotenkoordinaten in reale Weltkoordinaten zu transformieren oder man transformiert die Geometrien in Matrixkoordinaten. Da eine direkte Assoziation des Knotengitters mit der Geometrie softwaretechnologisch unangebracht ist, wurde das Verbindungselement - die *Interactor*-Klasse eingeführt. *Interactor*-Objekte beschreiben die Interaktion zwischen Hindernisstrukturen und dem Rechengitter. Sie verwalten die Geometrie, beschreiben physikalischen Eigenschaften (festes Hindernis, Geschwindigkeitsrand) und stellen die Wechselwirkung zwischen dem *NodeGrid* und der Hindernisgeometrie sicher. Das heisst, sobald sich die Geometrie ändert, muss das Knotengitter (*NodeGrid*) aktualisiert werden. Um beispielsweise die aus dem Knotengitter resultierenden Kräfte auf die Struktur zu berechnen, müssen die in Frage kommenden Knoten bekannt sein. Diese Randknoten sowie Knoten, die im Objekt liegen, werden im *Interactor* ermittelt. Somit wird bei bewegten Objekten nur die Liste der Randknoten und inneren Knoten durchzusucht, um veränderte Knotenzustände zu bestimmen.

7.3 Bewegte Geometrien

Die Art und Weise, mit der sich ein geometrisches Objekt durch das Rechengitter bewegt, wird in einem weiteren Modul gekapselt. Dies ist das sogenanntes *SteeringPlugin*, das die Methode *moveGeoObject()* implementiert und mit dem *Interactor* assoziiert ist. Die Bewegung eines Objektes kann z. B. eine Starrkörperbewegung (*RigidBodyMotion*) sein oder Kräfte können auf das Objekt übertragen werden und Verschiebungen über einen Strukturlöser berechnet werden. Hierzu sind mehrere *SteeringPlug-*

ins implementiert: zum einen die Klassen, die den Fluidlöser mit dem 1-D-FEM-Balkencode koppeln und zum anderen die Klassen, die mit dem 3-D-p-FEM Strukturlöser ADHOC koppeln. Hinzu kommt noch jeweils die Unterscheidung des Zeitintegrationsverfahrens in Eigenwert- und Newmarkverfahren. In Summe ergibt dies vier *SteeringPlugins*, die modular gewählt werden können. Wichtig ist, dass das zugehörige geometrische Objekt ein Polygon ist, das im Fluidgebiet den geometrischen Rand der verformbaren Stuktur beschreibt. Für den dreidimensionalen Strukturcode ist das Austauschnetz ein 3-D Oberflächennetz der Struktur. Dieses ist in Tiefenrichtung identisch, sodass ein Polygonquerschnitt mit dem 2-D Fluidlöser gekoppelt werden kann.

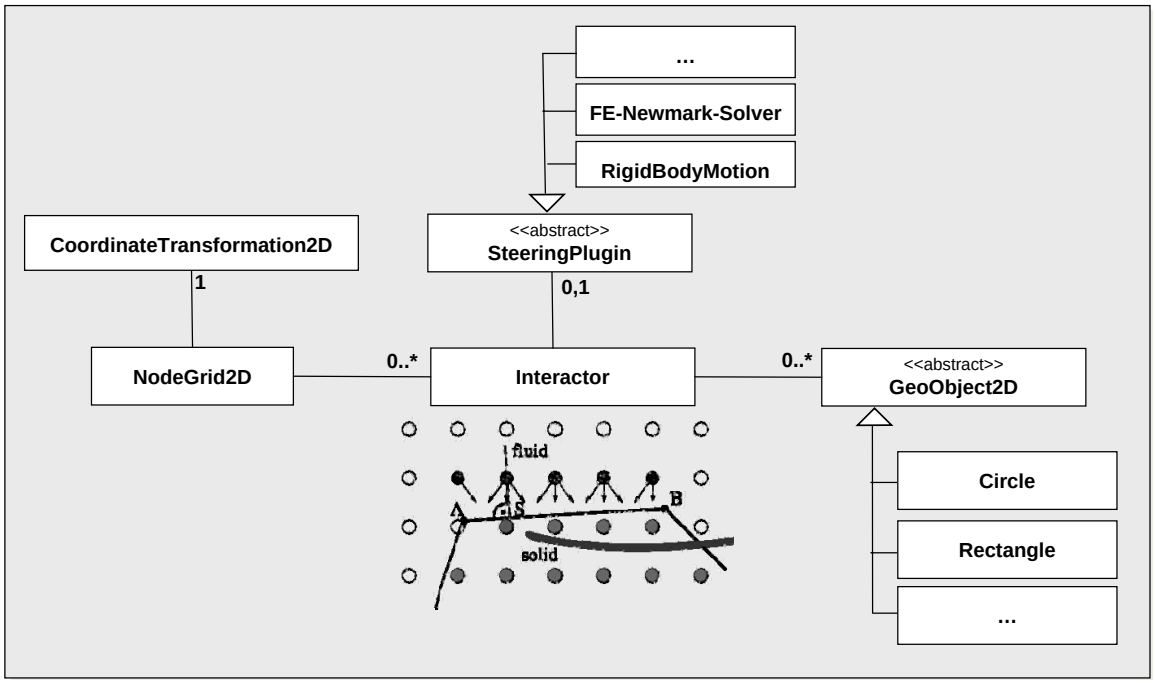


Abbildung 7.3: *UML-Diagramm für die Geometrie / Gittergenerierung*

7.4 Spezifische Gitter

Der *D2Q9Descriptor* ist ein spezifischer *NodeValueDescriptor* und das *D2Q9NodeGrid* ist ein spezifisches *NodeGrid*. Sie beschreiben für den Gittergenerierungsprozess notwendige Knotenwerte und Gittereigenschaften. Knotenwerte sind z. B. Skalierungsmarkierungen oder Randbedingungsmarkierungen. Das *D2Q9NodeGrid* definiert abstrakte Methoden für die spezielle D2Q9-Lattice-Boltzmann-Implementierung. Vom *D2Q9NodeGrid* und *D2Q9Descriptor* sind die speziellen physikalischen Klassen abgeleitet, wie z. B. für Einphasen-, Mehrphasen- und Flachwassersimulationen. Die jeweiligen Klassennamen kann man Abbildung 7.4 entnehmen. Die abgeleiteten Gitter haben einen spezifischen Rechenkern (*SedimentCalculator, OnePhaseCalculator, MultiPhaseCalculator, ...*) in dem die jeweilige Berechnung implementiert ist.

Ein Einphasenberechnungsgebiet hält im dazugehörigen *OnePhaseNodeValueDescriptor* die Verteilungen und im *OnePhaseNodeGrid* die Viskosität und die Volumenkräfte. Ein Knoten einer Flachwassersimulation wird mit dem *ShallowWaterNodeValueDescriptor* beschrieben. Er speichert zusätzlich zu den LB-Verteilungen die Oberflächenhöhe und die Bodenhöhe, die zur Berechnung notwendig sind. Das

zugehörige *ShallowWaterNodeGrid* hat neben der Gitterviskosität auch die Gravitation, die über das gesamte Rechengebiet konstant ist.

Wie man der Grafik entnehmen kann, stellt das Framework Klassen für verschiedene physikalische Kerne zur Verfügung. Idealerweise kann somit jeweils ein Anwender seine gegebene physikalische Fragestellung in seinem Unterpaket der Softwarestruktur behandeln. Die Gittergenerierung existiert eine Abstraktionsstufe höher und ist somit für alle Kerne gleich.

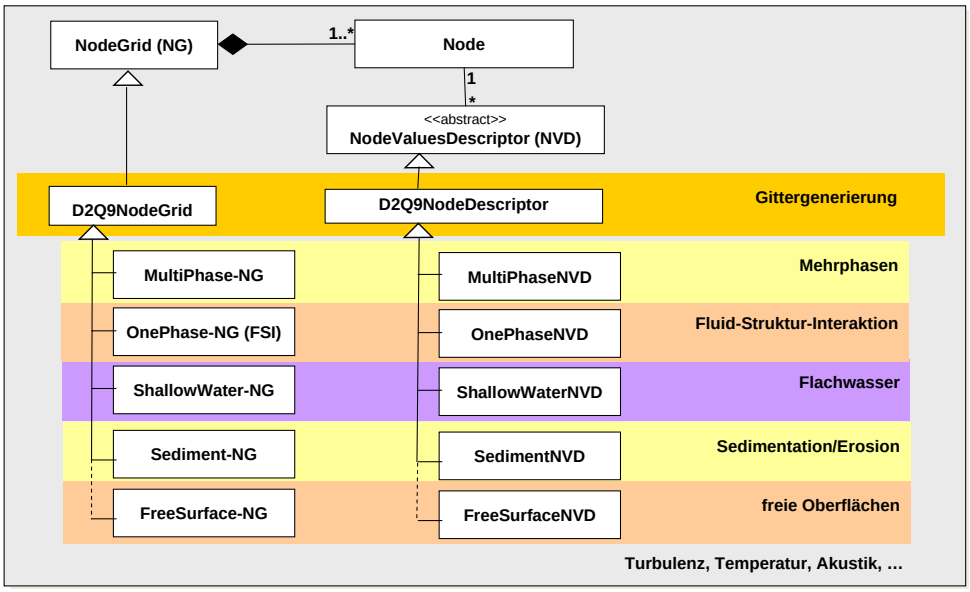


Abbildung 7.4: *UML-Diagramm für spezifische Rechengitter*

7.5 Instanziierung und Verwaltung von Objekten

Die Basisobjekte zur Instanziierung und Verwaltung sind, wie man in Abbildung 7.5 erkennt, das *ObObject*, der *ObObjectCreator*, der *ObObjectManager* und die *ObObjectFactory*. Das Kürzel *Ob* steht hierbei für das Paket *basics.objects* in dem sich die Klassen befinden Der Objektmanager (*ObObjectManager*) verwaltet die Objekte (*ObObject*) . Er besitzt Methoden zum Löschen, Hinzufügen sowie Editieren der Objekte. Die Objektfabrik (*ObObjectFactory*) ist eine Fabrik, in der Objekte erstellt werden. Sie verwaltet die Menge der einzelnen Maschinen bzw. objekterzeugenden Klassen (*ObObjectCreator*).

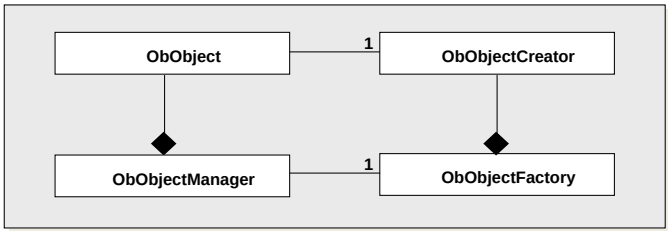


Abbildung 7.5: *generalisierte Manager-Factory-Komponenten*

Neben den geometrischen Objekten (*GeoObject2D*), den Knotengittern (*NodeGrid*) und Adaptionskriterien (*NodeAdaptationCriterion*) sind die Interaktoren (*Interactor*) eine spezifische Erweiterung dieser Klassen. Die Instanziierung von Interaktoren erfolgt mittels Erzeugern (*Creators*). Diese wurden im so genannten Singleton-Prinzip [35] (genau eine Instanz) implementiert. Die Menge der *Creators* wird in einer *InteractorFactory* gehalten, die ebenfalls ein Singleton ist. Auf die reale Welt übertragen bedeutet dies, dass es genau eine Fabrik (*InteractorFactory*) gibt, in der von jedem Typ genau eine Maschine (Creator) steht, die eine beliebige Anzahl von Produkten (*Interactors*) erstellen kann. Die Methode, die eine Instanz eines Objektes (*Interactor*) erstellt, ist der Definition zufolge eine Factory-Methode [35]. Auf diesem Weg werden beliebige Interaktionsobjekte erstellt, die in einem *InteractorManager* verwaltet werden. Für die Geometrien, Rechengitter und Adaptionskriterien gilt dies analog.

Diese abstrakte Darstellung der einzelnen Objekte ist vor allem zur persistenten Speicherung des Rechengitters notwendig. Bei der Fluid-Struktur-Interaktion möchte man oft einen eingeschwungenen Fluidzustand berechnen, bevor man das Setup mit bewegten Geometrien startet. Erzeugt man seinen Ausgangszustand von einem abgespeichertes Rechengitter, werden die jeweiligen Objekte über die einzelnen Erzeugenden (*Creators*) der Factory instanziiert. Normalerweise hätte man an dieser Stelle ein unübersichtliches if-then-Konstrukt, das schwer zu erweitern ist. Das Hinzufügen neuer Objekte und der Creatoren kann modular erfolgen und die Persistenz ist sichergestellt, sobald der Creator an der Factory angemeldet ist.

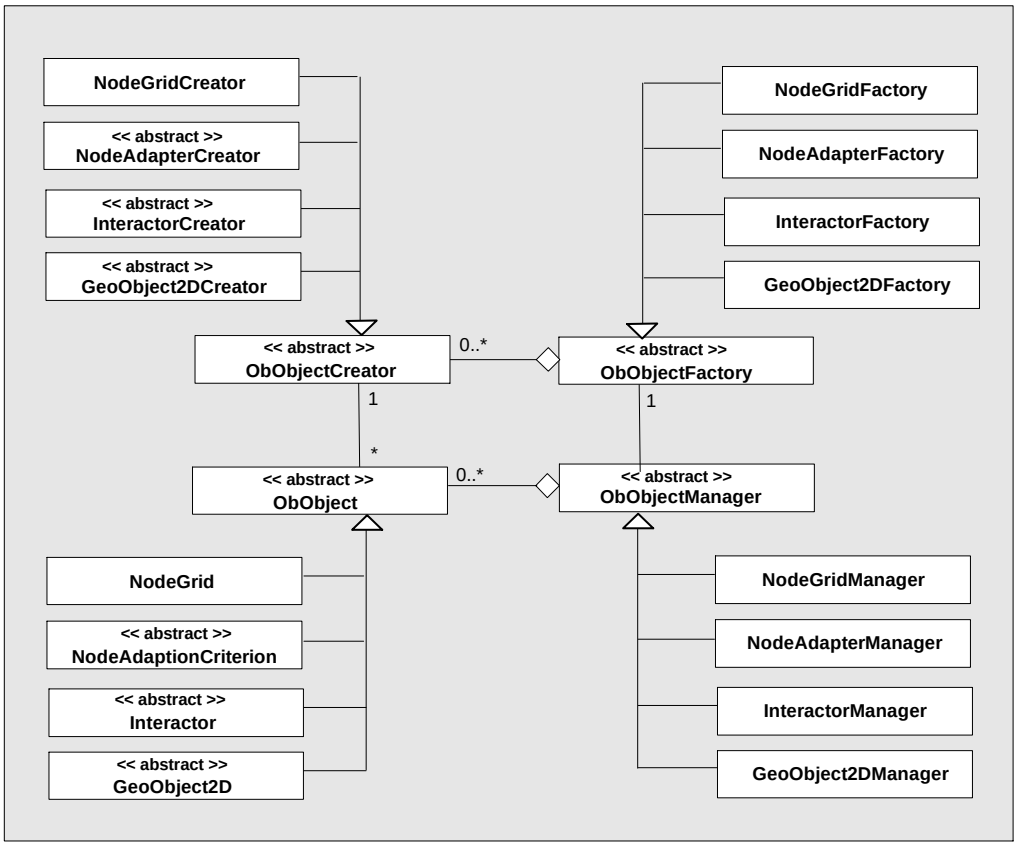


Abbildung 7.6: *UML-Diagramm für spezifische Manager-Factory-Komponenten*

7.6 Visualisierung

Zur interaktiven Visualisierung wird die Open-Source-Bibliothek Visualization ToolKit (VTK) der Firma Kiware [110, 64] genutzt. Das Visualization ToolKit ist ein Open-Source Softwaresystem für 3-D-Computergrafik, Bilderverarbeitung und Visualisierung. VTK liegt eine Ebene oberhalb von so genannten Renderingbibliotheken (in diesem Fall OpenGL [96]) und ist mit über 700 Klassen sehr umfangreich. Die in C++ geschriebene Bibliothek lässt sich auch unter Tcl [125], Python [99] und Java [57] verwenden. Das VTK bietet eine Vielzahl von Visualisierungsalgorithmen, die sowohl Skalar-, Vektor- und Tensordarstellung umfassen als auch fortgeschrittene geometrische Modellierungstechniken, wie Polygonreduktion oder Delaunaytriangulierung. VTK besteht im Wesentlichen aus einem kompilierten Kern und einer interpretierenden Schicht. Im kompilierten Kern sind Datenstrukturen, Algorithmen und vor allem zeitkritische Funktionen enthalten. Der Kern bzw. VTK ist unabhängig vom Graphical User Interface (GUI) und somit auch unabhängig vom Fenstersystem. Im Gegensatz zum Kern, bei dem es auf Geschwindigkeit und Effektivität ankommt, bietet die interpretierende Schicht Programmierkomfort und Flexibilität.

7.6.1 Grafikmodell

Das Grafikmodell erzeugt die Bilddaten und bildet eine abstrahierende Schicht über der verwendeten Standard-Grafikbibliotheken OpenGL [96]. Die Bezeichnungen der Objekte des Grafikmodells sind stark an Begriffe der Filmindustrie angelehnt. Mit Lichtern, Kameras, Darstellern und Requisiten, die ein Benutzer instanziiert und anordnet, entsteht eine Szene.

Es gibt sieben grundlegende Basisobjekte [110], die genutzt werden, um eine Szene zu rendern:

1. **vtkRenderWindow**: das Ausgabefenster, in das die Renderer zeichnen können.
2. **vtkRenderer**: koordiniert den Renderprozess mit Licht, Kamera und Darstellern.
3. **vtkLight**: eine Lichtquelle mit den Eigenschaften Position, Farbe, Helligkeit, etc.
4. **vtkCamera**: eine Kamera mit den Eigenschaften Position, Blickrichtung, etc.
5. **vtkActor**: ein beliebiges sichtbares Objekt mit den Eigenschaften Position, Transformation, etc.
6. **vtkProperty**: definiert die Erscheinungseigenschaften von *vtkActor*, wie Farbe und Transparenz, sowie Lichteigenschaften, wie diffus oder reflektierend.
7. **vtkMapper**: transformiert Daten, die durch andere vtk-Objekte generiert und modifiziert werden, in Geometrien

7.6.2 Visualisierungs-Pipeline

Zum Erstellen eines Visualisierungsstranges enthält VTK Klassen, die zum einen Datensätze erzeugen bzw. Informationen zwischenspeichern (Basisklasse *vtkDataObject*) und diese Daten verarbeiten können (Basisklasse *vtkProcessObject*). Die Daten werden über eine Rohrleitung (Pipeline) durch zahlreiche Filter geführt. Es gibt drei verschiedene Kategorien von datenverarbeitenden Prozessobjekten:

Quelle: liest Daten aus einer Datei ein oder berechnet die Daten algorithmisch. Diese werden dann in das Datenobjekt geschrieben. z. B.: *vtkConeSource*

Filter: transformiert Daten in ein neues Datenobjekt. z. B.: *vtkContourFilter*

Mapper: Die Abbildung zwischen den Visualisierungsdaten und den grafischen Daten wird von Mappern (*vtkMapper*) verrichtet. Er liest Daten aus dem Datenobjekt und macht sie für das Grafikmodell verfügbar, erstellt also geometrische Primitive. z. B.: *vtkPolyDataMapper*

Die Visualisierung folgt dem prinzipiellen Verfahren zur Trennung von topologischer Beschreibung und Präsentationsbeschreibung [10]. In diesem Fall transformiert ein *NodeGridPresentator*, die Daten zur Darstellung im *Viewer* (siehe Abbildung 7.7). Der *Viewer* ist eine generalisierte Komponente die ein *vtkRenderWindow* mit zugehörigem *vtkRenderer* verwaltet. Zusätzlich werden die Kameras *vtkCamera* und Lichter *vtkLight* im *Viewer* gehalten.

Die Presentatorklasse kapselt die Visualisierungspipeline und stellt am Ende ein *vtkActor*-Objekt für den *Viewer* bereit.

Diese Trennung war nötig, da die Berechnung unabhängig von der Grafik laufen soll, d.h. der Rechencode muss auch ohne die einzelnen Grafikbibliotheken kompilierbar und ausführbar sein. Wichtiger Bestandteil hierbei ist das Observer-Pattern [35]. Hierfür gibt es zwei Klassen *UbObserver* (Beobachterobjekt) und *UbObservable* (zu beobachtende Objekt). Ein observierbares Objekt ist von *UbObservable* abgeleitet oder implementiert dies als Assoziation. Für diese observierbaren Objekte können erstellte Beobachter (*UbObserver*) in einer Liste mit Beobachtern (*UbObserver*) gespeichert werden. Diese Liste wird bei jeder Änderung von Eigenschaften des Objektes aufgerufen.

Ist ein Objekt observierbar, verwaltet es eine Liste von Observern, die bei einer Änderung aktualisiert werden. Ein *NodeGrid* ist ein observierbares Objekt und von *UbObservable* abgeleitet. Die Presentatorkomponente assoziiert einen Beobachter (*UbObserver*), der in der Liste der Observer im NodeGrid gespeichert wird. Somit ist die Visualisierungsaktualisierung unabhängig von den Basisklassen.

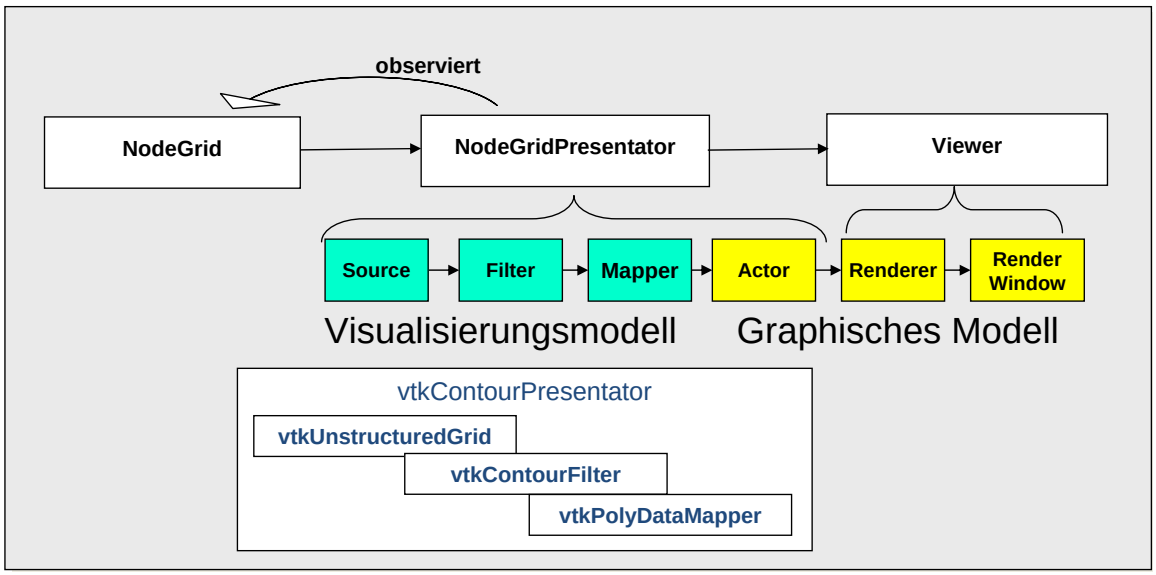


Abbildung 7.7: *Präsentationskomponenten*

7.7 GUI - Basiskomponenten

Die Visualisierung der Daten erfolgt parallel zu deren Berechnung. Dabei kann der Datenviewer als Desktopanwendung aber auch als immersives Betrachtungswerkzeug in einer Virtual-Reality-Umgebung mit aktiver Standorterfassung des Betrachters verwendet werden.

Die grafische Benutzeroberfläche ist folgendermaßen aufgebaut (vgl. Abbildung 7.8): Im oberen Bereich befindet sich die Menüleiste, links das Bedieninstrument und rechts der Datenbetrachter. Das Bedieninstrument zeigt im oberen Bereich die zu verwaltenden Komponenten und im unteren Bereich die spezifischen Objekteditoren (*SpecificObjectInstrument*). Wird die Simulation in einer interaktiven Umgebung ausgeführt, vergrößert man den Datenbetrachter zum Vollbildschirm und das Bedieninstrument erscheint als seperates Fenster.

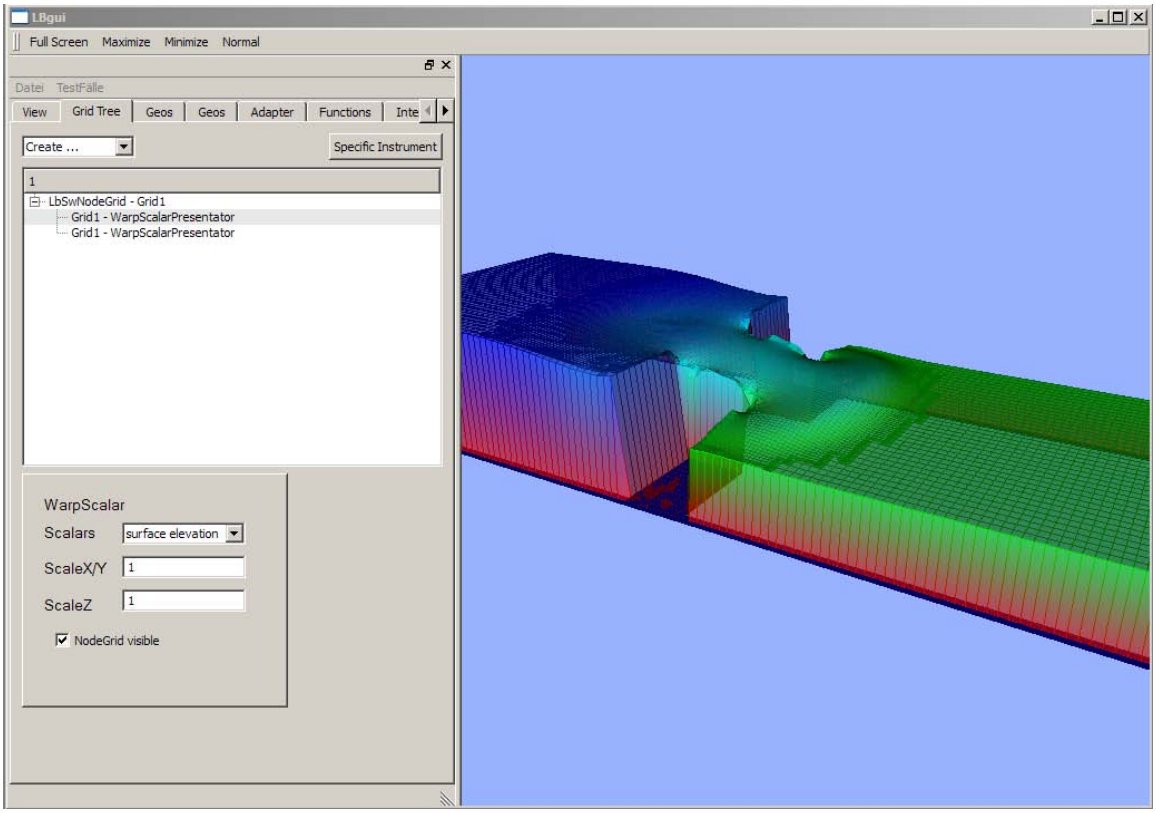


Abbildung 7.8: *Screenshot der interaktiven Anwendung*

Zur Darstellung auf dem Monitor besitzen alle Objekte bzw. Elemente spezifische Editoren. Der *ObObjectCreator* hat für das zu erzeugende Objekt ein spezifisches Instument, bei dem man die Parameter eingibt, die notwendig sind, um das spezifische Objekt zu erzeugen. Spezifische Objekte haben einerseits Editoren (*SpecificObjectInstrument*) zur Änderung ihrer quantitativen Eigenschaften und andererseits zugehörige Darstellungsobjekte (*Presentator*) zur qualitativen Darstellung z. B. in einer GUI. Editoren der Erzeugungsobjekte sind oftmals dieselben, die man auch für das Objekt selbst nutzt.

In Abbildung 7.9 sind die Basiskomponenten der grafischen Nutzerschnittstelle dargestellt. Man hat eine spezifische Applikation (*SpecificApplication*) die den Datenbetrachter beinhaltet. Weiterhin werden sogenannte *ManagerPresentatorInstrumente* assoziiert, die prinzipiell Panelfenster des Bedienbereichs

sind. Hier wird die jeweilige Objektliste sowie deren Darstellungskomponente (*Presentator*) dargestellt. Der *GeoObject2DManager*, der *InteractorManager*, der *NodeAdapterManager* sowie der *NodeGridManager* sind spezifische *ObjectManager*, die über ein *ManagerPresentatorInstrument* in der spezifischen Applikation dargestellt werden. Dieses Instrument beinhaltet neben der Liste der Objekte auch Zugriffsmöglichkeit auf die jeweiligen Darstellungskomponenten. Alternativ kann man das *ManagerInstrument* wählen, um nur die Objekte darzustellen.

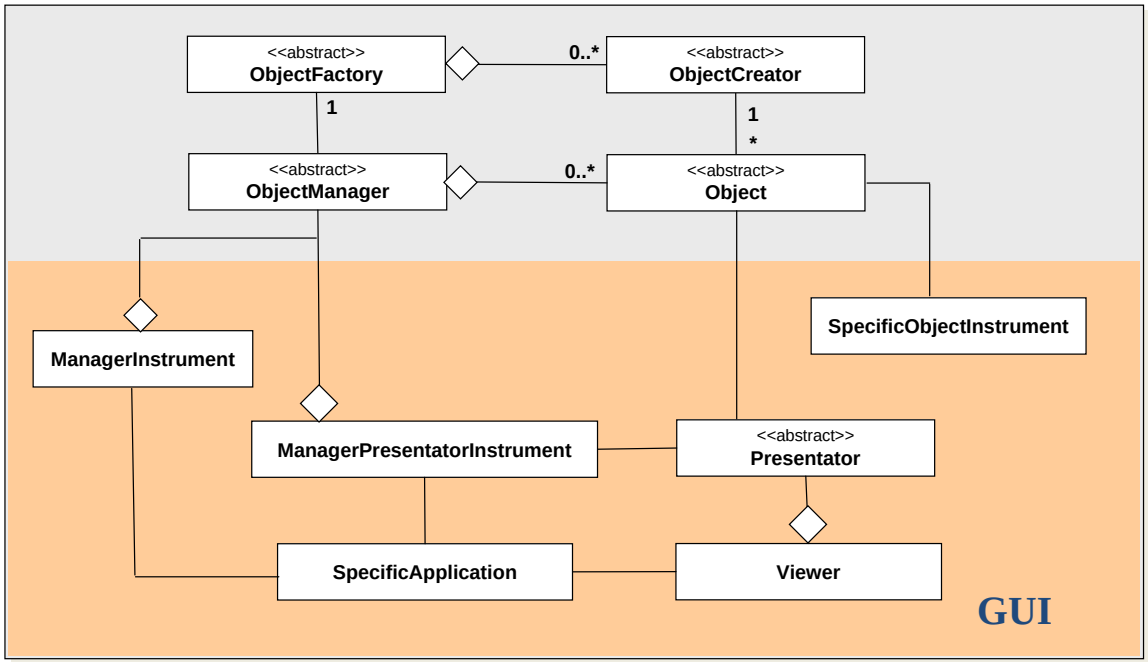


Abbildung 7.9: *GUI-Grundmodell*

7.8 Parallelisierung

Reale Probleme im Bauwesen sind oft mit turbulenten Strömungen assoziert und benötigen eine hohe Anzahl an Freiheitsgraden. Um die Berechnung in einer angemessenen Zeit durchzuführen, ist eine parallele Codeversion unabdingbar. Freudiger hat in seiner Arbeit [32] ein paralleles Softwareframework konzipiert und umgesetzt. Hier sollen kurz die wichtigsten Bausteine und die Erweiterung für FSI-Berechnungen beschrieben werden.

Zur Regelung des Datenaustausches in einer komplexen Anwendung zwischen verschiedenen Plattformen bzw. Prozessen nutzt man normalerweise Middlewareplattformen. Middleware wie CORBA [127] oder ICE [49] haben als Grundkonzept eine eigene Interface Definition Language (IDL) und sind mit dem Ziel der Sprachunabhängigkeit erstellt worden. Dies bedeutet für eine einfache Kommunikation zwischen Modulen, die mit derselben Programmiersprache erstellt wurden, einen beachtlichen Overhead vor allem auch im Design-Prozess, wenn man die Client-Server-Architektur auf das Serialisieren und Austauschen von Daten reduziert und nicht sprachübergreifend behandeln möchte. Java bietet mit RMI [121] eine einfache Schnittstelle zum Erzeugen von Interprozesskommunikation. In C++ muss man hier viele Teile händisch machen, wenn man z. B. serialisierte Objekte mit MPI [105] verschicken möchte und das einfache Starten von Prozessen, wie im MPI2 Standard beschrieben, arbeitet mit den gängigen MPI-Implementierungen nicht zufriedenstellend.

Das Remote Call Framework (RCF) [79] bietet hier eine Alternative und ist als Templatebibliothek ähnlich zur Standard Template Library (STL) zu nutzen. Man kann Komponenten als Threads oder Remote-Prozesse einfach ansteuern und hat dabei eine schlanke Grundstruktur.

Zur Parallelisierung wurde ein serviceorientiertes Framework mit Hilfe der RCF-Bibliothek erstellt. Es wird zwischen fünf verschiedenen Komponenten unterschieden:

1. *IPService*
2. *TopologyService*
3. *InteractorService*
4. *CalcManagerService*
5. *CalcService*

In Abbildung 7.10 sind ein *TopologyService* und zwei *CalcServices* dargestellt. Der *TopologyService* verwaltet die topologischen Informationen des Rechengitters und ist die Basis zur konsistenten Gitteranpassung auf den Clients. Funktionen zur gleichmäßigen Verteilung der Rechenlast sind ebenfalls in dieser Komponente. Im Berechnungsservice (*CalcService*) wird jeweils ein Teil des Rechengitters allokiert und beim Simulationsprozess berechnet. Der *CalcManagerService* steuert den Berechnungsablauf. Im *InteractorService* werden die zugehörigen Hindernisgeometrien und Randbedingungen sowie deren Bewegung durch Starrkörperbewegung oder Fluid-Struktur-Interaktion verwaltet. Man kann über Aufrufe des *InteractorService* Geometrien verändern und die notwendigen Funktionen werden bei den Rechenclienten (*CalcServices*) aufgerufen. Der *IPService* ist der zentrale Service, der die IP-Adressen ähnlich eines Nameservices der anderen Service verwaltet. Über ihn kann jeder Service feststellen, wo sich ein anderer Service befindet.

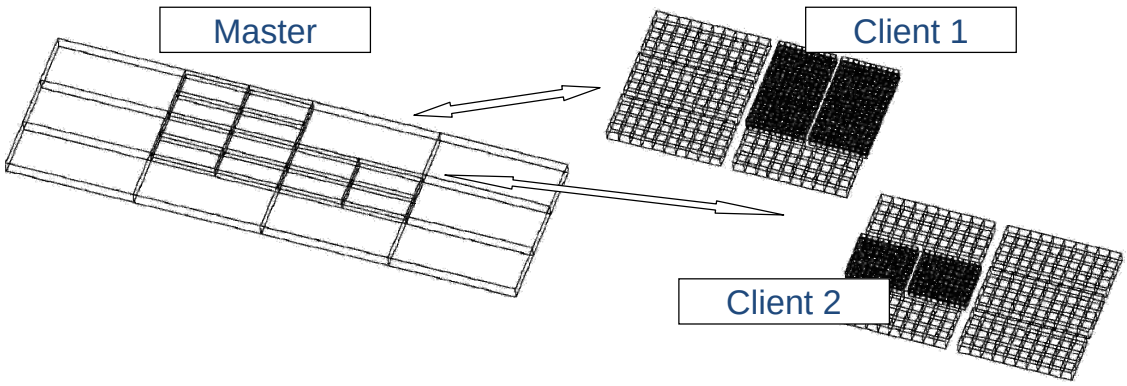


Abbildung 7.10: *hybride, verteilte Blockdatenstruktur*

Für die Fluid-Struktur-Interaktion wurde die Starrkörperengine PE [54] in einem zusätzlichen Service gekapselt (*PEService*). Dieser kommuniziert mit dem *InteractorService*, wodurch die Modifizierung der Geometrie stattfinden kann. Die bisherige Kopplung funktioniert für wenige im Gebiet stationierte Strukturobjekte. Um Systeme mit mehr als 10000 Starrkörpern effizient zu simulieren, muss der Strukturlöser parallel ausgeführt werden. Dies ist Gegenstand weiterer Implementierungsarbeiten.

7.9 Aspektorientierung (AOP)

Das Grundprinzip des numerischen Verfahrens für Einphasensimulation lässt sich im Wesentlichen durch zwei Methoden beschreiben: Kollision und Propagation. Für die Fluid-Struktur-Interaktion kommen Funktionen zur Berechnung der Bewegung der Struktur und der daraus resultierenden Rechengitteraktualisierung hinzu. Dies soll beispielhaft verdeutlichen, dass die Numerik und somit auch die Systemarchitektur der einzelnen Rechenkerne prinzipiell ähnlich ist. Man erhält automatisch Code, der bei den einzelnen Kernels redundant ist. Durch die immens gestiegene Komplexität der Ingenieuranwendungen ist selbst in der besten Softwarearchitekur das Problem des Code-Tangling nicht vermeidbar: Wenn Code, der verschiedene Anliegen betrifft, in einem Modul vermischt ist, wird dies als Code-Tangling bezeichnet. Abhilfe schafft hier die aspektorientierte Programmierung, die in der C++-Erweiterung AspectC++ [2, 117] umgesetzt wurde. Diese bietet dem Entwickler eine Alternative, um die genannten Probleme zu umgehen.

Mit der aspektorientierten Programmierung wird es möglich, die so genannten querschneidenden Belange (Crosscutting Concerns) in modularer Weise zu implementieren. Dies sind Belange in der Softwareentwicklung, die zu Programmcode führen, der nicht in das Modularisierungsschema des restlichen Programms passt, und so über weite Teile des Programms verstreut wird. Statt einen logisch zusammenhängenden Programmcode in Fragmente zu zerlegen und an verschiedenen Stellen eines Programms einzusetzen, wird die implizite Ausführung genutzt. Die Module, die dies leisten, werden Aspekte genannt und bestehen aus den auszuführenden Codefragmenten selbst und einer Beschreibung, von welchen Punkten aus die Ausführung stattfinden soll. Ein solcher Punkt wird als Join Point bezeichnet.

Den technischen Vorgang, der für die implizite Ausführung des Aspektcodes an den Join Points sorgt, nennt man Aspektweben. Der Aspektweber kann zum Beispiel ein Codetransformationssystem sein, das Funktionsaufrufe im Präprozess generiert, ein Compiler, der entsprechende Aufrufe als Maschineninstruktionen einfügt oder ein Laufzeitsystem, das in einem Interpreter oder eine virtuelle Maschine eingebunden ist und dort Verbindungspunkte zur Laufzeit erkennt und mit Aspektcode verbindet.

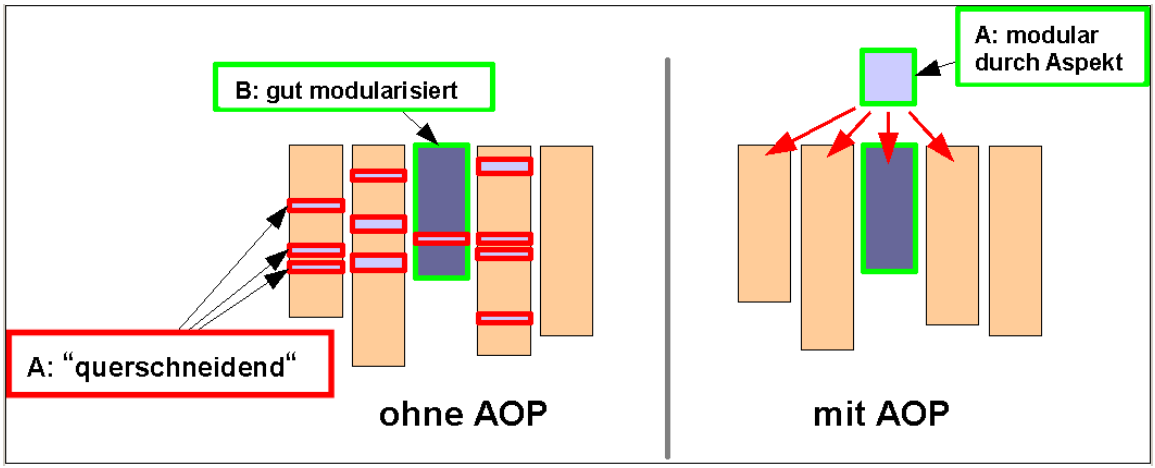


Abbildung 7.11: *Modulare Implementierung eines querschneidenen Belanges [116]*

Abbildung 7.11 zeigt links die Implementierung eines querschneidenen Belanges ohne und rechts mit Hilfe von Aspekten. Die Rechtecke symbolisieren die verschiedenen Quelltexteinheiten, die zu dem Programmsystem gehören. Der Code, der für die Implementierung des querschneidenen Belanges ”A” benötigt wird, ist hervorgehoben. Wie zu erkennen ist, führt die Lösung mit Aspekten auf der rechten

Seite zu einer verbesserten Modularität. Auf diese Weise wird durch die Anwendung von Aspekten die Lesbarkeit des Codes erhöht und damit seine Wartbarkeit verbessert. Zudem wird die zu implementierende Codemenge reduziert, da redundanter Code vermieden und somit die Produktivität der Entwickler gesteigert wird.

Abbildung 7.12 zeigt examplarisch die Zeitmessung zur Geschwindigkeitsoptimierung der einzelnen Funktionen zweier numerischer Rechenkerne. Bedingte Übersetzung und Code, der für die eigentliche Berechnung nicht benötigt wird, macht das Verstehen der Abläufe schwierig.

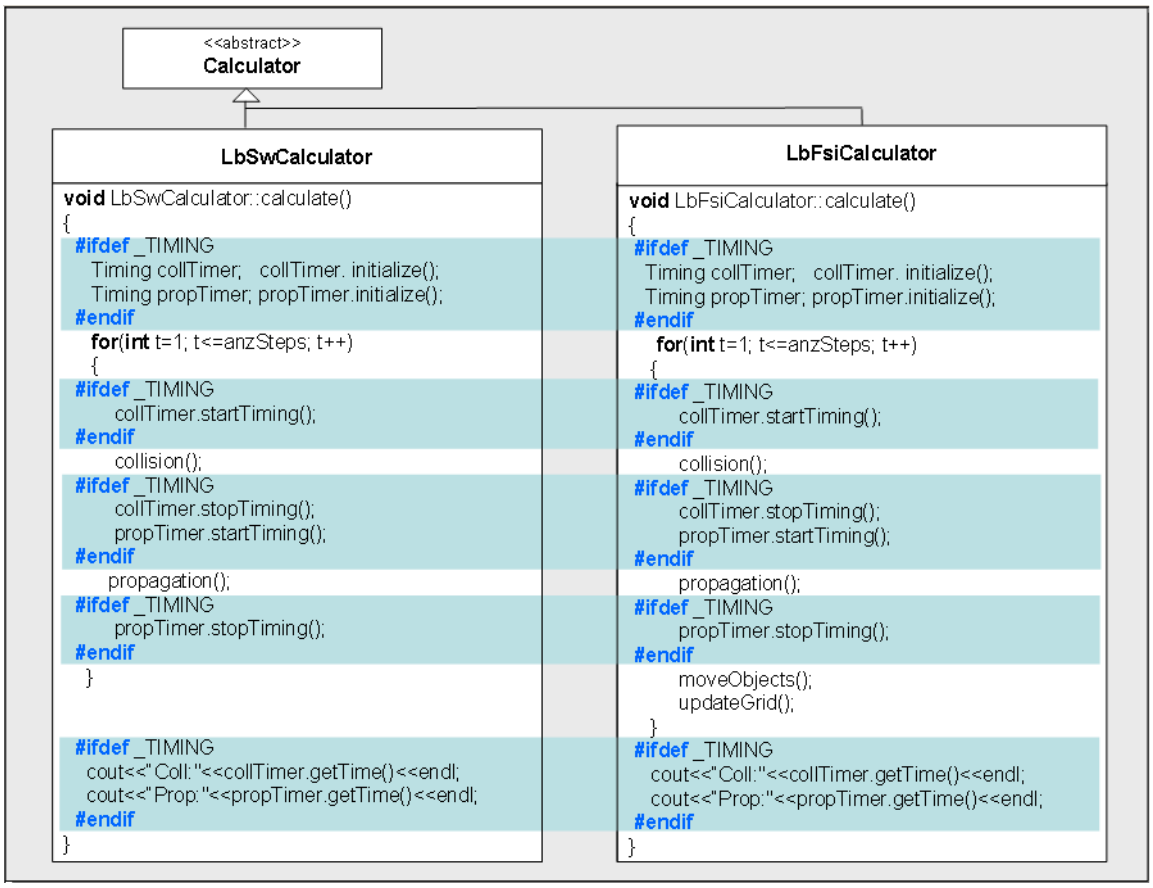


Abbildung 7.12: *Zeitmessung ohne AOP-Code*

Abbildung 7.13 zeigt dieselben Funktionen der Berechnungsklassen unter Verwendung eines Aspekts. Hier bleibt lediglich der wesentliche Kernbestandteil bestehen. Der zusätzliche Code für den Aspekt *TimeMeasurement* ist auf der linken Seite der Abbildung dargestellt und mit Kenntnis der wenigen Grundelemente von AspectC++ schnell nachvollziehbar. Wie man sehen kann, befindet sich bei dieser Implementierungsvariante kein Code zur Zeitmessung mehr in den Klassen *LbSwCalculator* und *LbFsiCalculator*. Dieser ist nun einzig im Aspekt *TimeMeasurement* zu finden, was die Gesamtlänge des Codes reduziert und die Duplikation vermeidet. Darüber hinaus wurde die Zahl der #ifdef Direktiven im Code von zehn auf eins verringert. All dies fördert nicht nur die Lesbarkeit sondern auch die Austauschbarkeit der Module. Der Code den ein Mitarbeiter in seinem Forschungszweig (assoziert mit einem numerischen Kern) erstellt hat, wird an den entsprechenden Stellen eingewoben und erspart insbesondere Codeumbauzeit. Oftmals hat man den Effekt, dass neue Module in einem Forschungscode fehlerfrei arbeiten, in

anderen Projekten jedoch nicht. Der dann notwendige Um- und Rückbau des Codes ist normalerweise sehr zeitintensiv. Das Einweben neuer Module spart hier viel Entwicklungszeit.

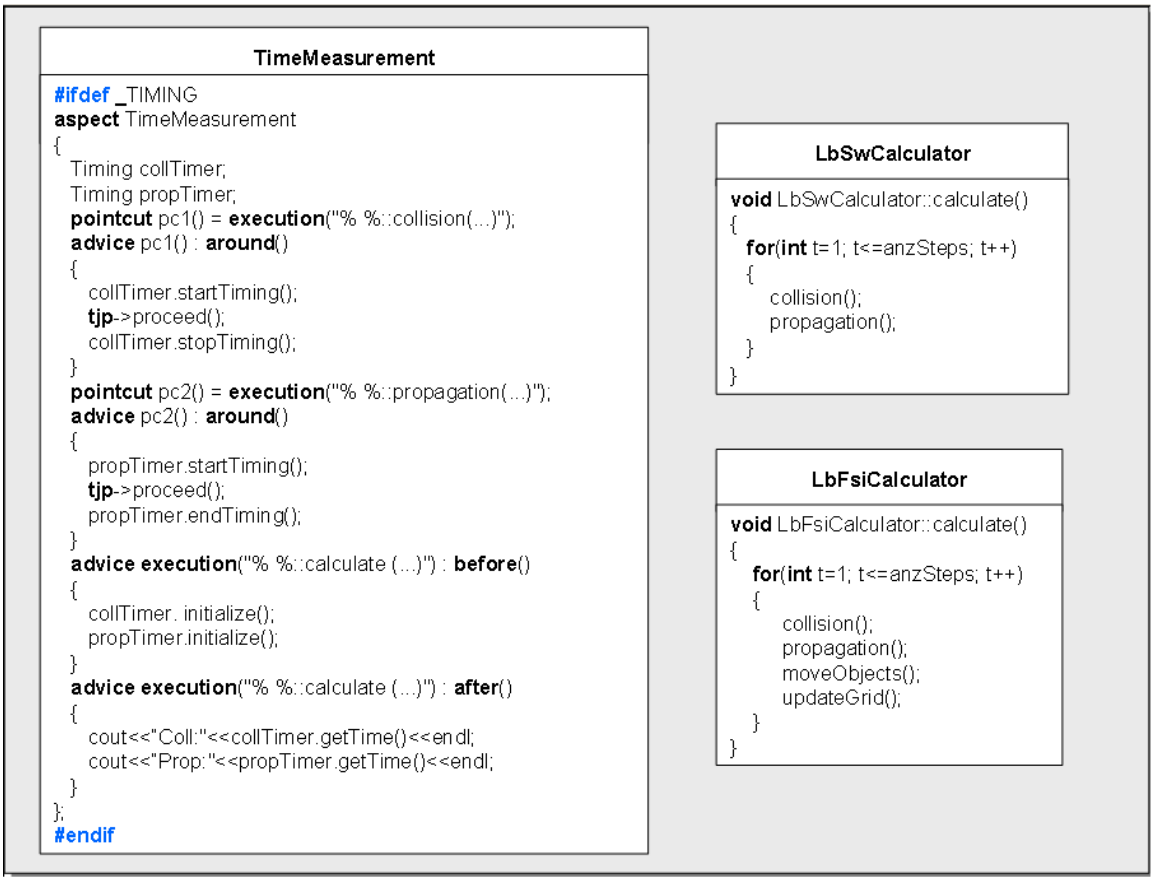


Abbildung 7.13: *Zeitmessung mit AOP-Code*

7.10 Weiterentwicklung

Angetrieben durch die komplexe Zielstellung wird die VIRTUALFLUIDS-Softwarebibliothek stetig weiter entwickelt. In absehbarer Zukunft wird es möglich sein, alternativ zu teuren Wellenkanalexperimenten und Windkanalexperimenten numerisch eine Vielzahl von Szenarien durchzuspielen. Dabei kann der Modellierer Parameter beliebig anpassen, ohne eine komplette Neuorganisation des Experimentes vornehmen zu müssen. Ziel ist es, sowohl kleine Probleme ohne große Kosten und Aufwand zu berechnen, als auch reale Szenarien simulieren zu können.

Da mit wachsenden Modell- und Rechengebietsgrößen auch die Menge der zu visualisierenden Daten zunimmt, muss für die Betrachtung eine Parallelisierung des Renderings realisiert werden. Durch geeignete Algorithmen kann alternativ die Datenmenge reduziert werden.

Im VTK-Paket sind Ansätze für das parallele Rendern von Daten enthalten, jedoch sind diese bisher noch nicht in die Plattform eingebunden. Angestrebt wird eine Umsetzung, bei der der parallele Rechenkern seine Daten direkt an einen parallelen Renderer liefert. Der Renderer wird dabei mit Informationen von der Anwendungsplattform über Standpunkt des Beobachters und dargestellten Bildausschnitt versorgt.

Aus den Simulations- und Betrachterinformationen wird dann ein Bild berechnet, das in komprimierter Form vom Rechencluster an den Viewerclienten verschickt werden kann. Der Vorteil einer solchen Lösung liegt in der geringen Netzlast und der beliebigen Anwendbarkeit auf verschiedensten Rechnern und Betriebssystemen.

Ohne das in diesem Kapitel beschriebene Softwaremodell wäre das Ziel der Umsetzung der Komplexität der geforderten Anwendung nur schwer möglich. Es ist bei geringem Implementierungsaufwand möglich, das Framwork für neue Modelle, wie z. B. ein thermisches Modell, zu erweitern. Eine Architektur mit verschiedenen Rechenkernen in einem Rechenprozess ist ebenfalls wünschenswert. Es ist vorstellbar, dass die zweidimensionale Flachwassersimulation direkt mit der dreidimensionalen freien Oberflächensimulation gekoppelt wird. Oder man könnte, um Rechenzeit und Speicherbedarf zu sparen, eine Einphasensimulation mit einer Mehrphasensimulation für die betreffenden Gebiete koppeln. Hierzu sind weitere Anstrengungen beim Designprozess nötig.

8 Schlußfolgerungen und Ausblick

In der vorliegenden Arbeit wurde ein Ansatz zur partitionierten Lösung gekoppelter bidirektionaler Fluid-Struktur-Probleme untersucht. Die Wahl der Löser wurde durch ihre Effizienz und Genauigkeit innerhalb ihrer Bereiche motiviert. Die unterschiedliche Diskretisierung des Fluids verglichen mit der Struktur ruft eine spezielle Handhabung des Transfers der Verschiebungen und Kräfte hervor. Die vorgeschlagenen Methoden erfüllen die Übergangsbedingungen am Interface.

Mit dem Benchmark für poröse Medien konnte gezeigt werden, dass VIRTUALFLUIDS konkurrenzfähig zu Lösern ist, die auf dem neuesten Stand der Technik sind. Die Beispiele legen nahe, dass für den schwach kompressiblen Fall der Lattice-Boltzmann-Ansatz einen signifikanten Zeitvorteil verglichen mit klassichen Methoden wie die Diskretisierung der Navier-Stokes-Gleichung mit Finiten-Volumen oder Finiten-Elementen hat.

VIRTUALFLUIDS wurde in 2-D in Bezug auf Fluid-Struktur-Interaktion ausführlich validiert. Es wurden mit einem Eigenwertlöser, bei dem man nur die niedrigsten Eigenmoden berücksichtigt, Ergebnisse derselben Größenordnung ermittelt. Hiermit konnte man aber kein geometrisch nichtlineares Verhalten, wie im Benchmark vorgegeben, nachbilden. Durch die Justierung der Parameter beim Newmarkverfahren lieferten die nichtlinearen Rechenfälle stabile Ergebnisse. Diese stimmten beim numerischen Benchmark sehr gut mit den Referenzwerten überein. Die vorgeschlagenen Methoden zum Transfer der Verschiebungen und Lasten sind somit geeignet. Beim experimentellen Benchmark wurde eine Genauigkeit von 20 Prozent erreicht, wodurch eine zusätzliche Betrachtung des Testfalles motiviert wird. Verbesserte Randbedingungen, Lastintegrationsalgorithmen sowie Prediktor-Verfahren zur Kopplung sollten daher untersucht werden. Weitere Modellfehler sind die Annahme eines quasi zweidimensionalen Verhaltens des dreidimensionalen Experimentes. Nötige Konvergenzstudien sind numerisch sehr aufwändig.

In drei Raumdimensionen wurde gezeigt, dass der Fluidlöser korrekte Ergebnisse liefert. Mit der sinkenden Kugel wurde ein Interaktionstestfall für bewegte Starrkörper untersucht, der ein Prozent Genauigkeit in Bezug auf die Abschätzformel aufwies. Die Kopplung mit der verteilt rechnenden, servicebasierten Codeversion des VIRTUALFLUIDS-Strömungslösers konnte ebenfalls an diesem Testfall validiert werden.

Der von Bathe und Ledezma vorgeschlagene Fluid-Struktur-Interaktionsbenchmark einer längsangeströmten Rechteckplatte wurde genutzt, um die Kopplung mit dem p-FEM-Strukturlöser ADHOC zu überprüfen. Die Ergebnisse weichen ca. 30 Prozent von den Vergleichswerten ab. Weitere Validierungen mit anderen FSI-Lösern und Konvergenzstudien sind nötig, um eine genauere Aussage über die Ergebnisse zu treffen.

Im Rahmen dieser Arbeit ist es gelungen, ein stabiles Simulationsverfahren in 2-D und 3-D zu entwickeln, mit dem Berechnungen für große Strukturverformungen möglich sind. Der hierzu notwendige Gittergenerator kann bewegte, komplexe Geometrien auf quad-/octreeartigen Gittern in einer zeitabhängigen Berechnung behandeln. Die Verfahren zur Punkt-in-Objekt-Bestimmung müssen weiter geprüft werden, da einige Sonderfälle nicht korrekt abgebildet werden. Die Effizienz kann durch Algorithmen, wie z. B. der Speicherung der Dreiecke in einer Baumstruktur zur Reduzierung der Komplexität beim Ray-Crossing-Algorithmus, verbessert werden. Zur Simulation von realistischen Strömungsphänomenen müssen die bekannten Turbulenzmodelle und die Gültigkeit der Kopplungsalgorithmen im Kontext der Fluid-Struktur-Interaktion untersucht werden. Hierfür ist es unabdingbar, dass die Simulationen verteilt ablaufen.

Weiterhin beinhaltet die dreidimensionale Codeversion ein freies Oberflächenmodell. Hier kann die Fluid-Struktur-Kopplung für Bauwerke, wie z. B. Offshorewindanlagen oder Brücken untersucht wer-

den. Das hierfür entwickelte Softwareframework bietet die Möglichkeit, neue numerische Modelle einzubauen und diese mit beweglichen Elementen zu koppeln.

Der Prototyp der interaktiven Simulationsumgebung läuft u. a. in der Virtual-Reality-Umgebung des Institutes für rechnergestützte Modellierung im Bauingenieurwesen. Zur Darstellung großer Datenmengen ist es wichtig, den Viewer als Serviceanwendung zu starten und ihn dynamisch an eine laufende Berechnung anzukoppeln. Somit soll es dann auch möglich sein, sehr große Datensätze zu untersuchen und interaktiv auf Rechenparameter in einer verteilt rechnenden Simulation Einfluss zu nehmen. Insgesamt erscheint der gezeigte Ansatz vielversprechend, um eine komplexe Simulationsumgebung zu entwickeln, die den Ansprüchen von verteiltem Rechnen und wissenschaftlicher Visualisierung gerecht wird. Dies bleibt im Hinblick auf neueste Hardwareentwicklungen eine Herausforderung.

Literatur

[1] *Static Object Intersections*. http://www.realtimerendering.com/intersections.html. [Online; zugegriffen 23-Juli-2009].

[2] AspectC++. http://www.aspectc.org/. [Online; zugegriffen 25-August-2008].

[3] Bathe, K. J. und Ledezma, G.: *Benchmark problems for incompressible fluid flows with structural interactions*. Computers & Structures, 85 (11-14):628–644, 2007.

[4] Bernsdorf, J., Durst, F. und Schäfer, M.: *Comparison of Cellular Automata and Finite Volume Techniques for simulation of incompressible Flow in Complex Geometries*. Int. J. Num. Meth. Fluids, 29:251–264, 1999.

[5] Bhatnagar, P. L., Gross, E. P. und Krook, M.: *A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems*. Physical Review, 94:511–525, Mai 1954.

[6] Blum, N.: *Algorithmen und Datenstrukturen: Eine anwendungsorientierte Einführung*. Oldenbourg Wissenschaftsverlag, Oldenbourg, 2004.

[7] Bouzidi, M., Firdaouss, M. und Lallemand P.: *Momentum transfer of a lattice-Boltzmann fluid with boundaries*. Physics of Fluids, 13:3452–3459, 2001.

[8] Brenk, M., Bungartz, H. J., Mehl, M. und Neckel, T.: *Fluid-Structure Interaction on Cartesian Grids: Flow Simulation and Coupling Environment*. In: Bungartz, H. und Schäfer, M. (Hrsg.): *Fluid-Structure Interaction, Modelling, Simulation and Optimisation*, Bd. 53 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 270–293. Springer, 2006, ISBN 978-3540345954.

[9] Brüggemann, B. und Holz, K. P.: *A Toolkit for Quadtree based Adaptive Simulation Systems - Software Engineering in Finite Volume Shallow Water Modeling*. In: *6th International Conference on Hydroscience and Engineering*, Brisbane, 2004. ISBN 0-937099-12-0.

[10] Brüggemann, M.: *Dynamische Interaktive Technische Dokumente*. Doktorarbeit, Lehrstuhl für Bauinformatik, Fakultät Architektur, Bauingenieurwesen und Stadtplanung, BTU Cottbus, 2000, ISBN 978-3-934934-02-3.

[11] Brüggemann, M.: *Informationsmodellierung im Bauwesen*. Postdoctoral thesis, Lehrstuhl für Bauinformatik, Fakultät Architektur, Bauingenieurwesen und Stadtplanung, BTU Cottbus, 2007, ISBN 978-3-934934-12-2.

[12] Carvaiho, P. und Cavalcanti, P.: *Point in Polyhedron Testing Using Spherical Polygons*. Graphics Gems V edited by Alan Paeth, S. 42–49, 1995.

[13] Cebral, J. und Löhner, R.: *On the loose coupling of implicit time-marching codes*. In 43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers, S. 10079–10093, 2005.

[14] CFX, A. http://www.ansys.com/products/cfx.asp. [Online; zugegriffen 10-Juli-2008].

[15] Chapman, S. und Cowling, T.: *The Mathematical Theory of Non-Uniform Gases*. Cambridge University Press, New York, 1970.

[16] Chung, J. und Hulbert, G.: *A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation: The Generalized-α-Method*. J. of Applied Mechanics, 60:1562–1566, 1993.

[17] Crouse, B.: *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen*. Dissertation, TU München, 2003.

[18] Crouse, B., Rank, E., Krafczyk, M. und Tölke, J.: *A Lb-Based Approach for Adaptive Flow Simulations*. International Journal of Modern Physics B, 17:109–112, 2003.

[19] d'Humières, D.: *Generalized lattice-Boltzmann equations*. In: Shizgal, B. D. und Weave, D. P. (Hrsg.): *Rarefied Gas Dynamics: Theory and Simulations*, Bd. 159 d. Reihe *Prog. Astronaut. Aeronaut.*, S. 450–458, Washington DC, 1992. AIAA.

[20] d'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P. und Luo, L. S.: *Multiple-relaxation-time lattice Boltzmann models in three dimensions*. Royal Society of London Philosophical Transactions Series A, 360:437–451, März 2002.

[21] Düster, A.: *High-Order Finite Elements for Three-Dimensional, Thin-Walled Nonlinear Continua*. Dissertation, TU München, 2001. http://www.inf.bv.tum.de/$\sim$duester.

[22] Düster, A., Bröker, H., Heidkamp, H., Heißerer, U., Kollmannsberger, S., Krause, R., Muthler, A., Niggl, A., Nübel, V., Rücker, M. und D.Scholz: *AdhoC*^{4} *– User's Guide*. Lehrstuhl für Bauinformatik, Technische Universität München, 2004.

[23] Düster, A., Bröker, H. und Rank, E.: *The p-version of the finite element method for three-dimensional curved thin walled structures*. Int. J. Num. Meth. Eng., 52:673–703, 2001.

[24] EXA. http://www.exa.com/. [Online; zugegriffen 21-Januar-2009].

[25] Fayon, A. und Happel, J.: *Effect of a cylindrical boundary on fixed rigid sphere in a moving fluid*. AIChE, 6(1):55–58, 1960.

[26] Featflow. http://www.featflow.de. [Online; zugegriffen 10-July-2008].

[27] Feng, Y., Han, K. und Owen, D.: *Coupled lattice Boltzmann method and discrete element modelling of particle transport in turbulent fluid flows: Computational issues*. International Journal for Numerical Methods in Engineering, 72(9):1111–1134, 2007.

[28] Feng, Y. und Michaelides, E.: *The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems*. Journal of Computational Physics, 195(2):602–628, 2004.

[29] Filippova, O. und Hänel, D.: *Boundary-Fitting and Local Grid Refinement for Lattice-BGK Models*. International Journal of Modern Physics C, 9:1271–1279, 1998.

[30] Förster, C., Wall, W. und Ramm, E.: *Artificial Added Mass Instabilities in Sequential Staggered Coupling of Nonlinear Structures and Incompressible Flows*. Computer Methods in Applied Mechanics and Engineering, 196:1278–1293, 2007.

[31] Freudiger, S.: *Effiziente Datenstrukturen für Lattice-Boltzmann-Simulationen in der computergestützten Strömungsmechanik*. Diplomarbeit, 2001.

[32] Freudiger, S.: *Entwicklung eines parallelen, adaptiven, komponentenbasierten Strömungskerns für hierarchische Gitter auf Basis des Lattice Boltzmann Verfahrens*. Dissertation, TU Braunschweig, 2009.

[33] Freudiger, S., Hegewald, J. und Krafczyk, M.: *A parallelization concept for a multi-physics lattice Boltzmann solver based on hierarchical grids*. Progress in Computational Fluid Dynamics, 8(1-4):168–178, 2008.

[34] Frisch, U., d'Humières, D., Hasslacher, B., Lallemand, P., Pomeau, Y. und Rivet, J. P.: *Lattice Gas Hydrodynamics in Two and Three Dimensions*. Complex Systems 1, S. 75–136, 1987.

[35] Gamma, E., Helm, R., Johnson, R. und Vlissides, J.: *Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software*. Addison Wesley Verlag, 2001.

[36] Geller, S.: *Object-Oriented Modelling of an Adaptive, Quadtree-based Finite Volume Method for the Shallow Water Equations*. Diplomarbeit, 2002.

[37] Geller, S., Krafczyk, M., Tölke, J., Turek, S. und Hron, J.: *Benchmark computations based on Lattice-Boltzmann, Finite Element and Finite Volume Methods for laminar Flows*. Computers & Fluids, 35:888–897, Nov. 2006.

[38] Geller, S., Tölke, J. und Krafczyk, M.: *Lattice Boltzmann Methods on Quadree-Type Grids for Fluid-Structure Interaction*. In: Bungartz, H. und Schäfer, M. (Hrsg.): *Fluid-Structure Interaction, Modelling, Simulation and Optimisation*, Bd. 53 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 270–293. Springer, 2006, ISBN 978-3540345954.

[39] Gerstenberger, A. und Wall, W.: *An eXtended Finite Element Method / Lagrange Multiplier based approach for fluid-structure interaction*. Computer Methods in Applied Mechanics and Engineering, 197:1699–1714, 2008.

[40] Ginzburg, I. und d'Humières, D.: *Multi-reflection boundary conditions for lattice Boltzmann models*. Physical Review E, 68:066614.1–066614.30, 2003.

[41] Ginzburg, I., Verhaeghe, F. und d'Humières, D.: *Two-Relaxation-Time Lattice Boltzmann Scheme: About Parametrization, Velocity, Pressure and Mixed Boundary Conditions*. Commun. Comput. Physics, 3(2):427–478, 2008.

[42] Gomes, J. und Lienhart, H.: *Experimental study in a fluid-structure interaction reference test case*. In: Bungartz, H. und Schäfer, M. (Hrsg.): *Fluid-Structure Interaction, Modelling, Simulation and Optimisation*, Bd. 53 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 356–370. Springer, 2006, ISBN 978-3540345954.

[43] Haines, E.: *Point in Polygon Strategies*. Graphics Gems `IV`, S. 24–46, 1994.

[44] Han, K. andFeng, Y. und Owen, D.: *Coupled lattice Boltzmann and discrete element modelling of fluid-particle interaction problems*. Computers & Structures, 85:1080–1088, 2007.

[45] Hänel, D.: *Molekulare Gasdynamik - Einführung in die kinetische Theorie der Gase und Lattice-Boltzmann-Methoden*. Springer-Verlag, Berlin, Heidelberg, 2004, ISBN 3-540-44247-2.

[46] Hashtabelle. `http://de.wikipedia.org/wiki/Hashtabelle`. [Online; zugegriffen 17-September-2009].

[47] He, X. und Luo, L. S.: *Lattice Boltzmann model for the incompressible Navier-Stokes equation*. Journal of Statistical Physics, 88(3-4):927–944, Aug. 1997, ISSN 0022-4715.

[48] He, X. und Luo, L. S.: *Theory of the lattice Boltzmann method: from the Boltzmann equation to the lattice Boltzmann equation*. Phys. Rev. E, 56:6811, 1997.

[49] Henning, M. und Spruiell, M.: *Distributed Programming with Ice*. `http://zeroc.com/download/Ice/3.3/Ice-3.3.0.pdf`. [Online; zugegriffen 10-Juli-2008].

[50] Hron, J. und Turek, S.: *A monolithic FEM solver for ALE formulation of fluid structure interaction with configurations for numerical benchmarking*. In: Papadrakakis, M., Onate, E. und Schrefler, B. (Hrsg.): *Computational Methods for Coupled Problems in Science and Engineering, Konferenzband First International Conference on Computational Methods for Coupled Problems in Science and Engineering, Santorini*, Bd. 1, S. 148. 2005.

[51] Hron, J. und Turek, S.: *Proposal for numerical benchmarking of fluid-structure interaction between elastic object and laminar incompressible flow*. In: Bungartz, H. und Schäfer, M. (Hrsg.): *Fluid-Structure Interaction, Modelling, Simulation and Optimisation*, Bd. 53 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 371–385. Springer, 2006, ISBN 978-3540345954.

[52] Hübner, B., Walhorn, E. und Dinkler, D.: *A monolithic approach to fluid-structure interaction using space-time finite elements*. Computer Methods in Applied Mechanics and Engineering, 193(23–26):2087–2104, 2004.

[53] Idelsohn, S., Oñate, E., Del Pin, F. und Calvo, N.: *Fluid-structure interaction using the particle finite element method*. Computer Methods in Applied Mechanics and Engineering, 195(17-18):2100–2123, 2006.

[54] Iglberger, K.: *pe - physics engine*. http://www10.informatik.uni-erlangen.de/~klaus/, 2008. [Online; zugegriffen 23-April-2009].

[55] Izquierdo, S. und Fueyo, N.

[56] Janßen, C.: *Free surface tracking with the Lattice Boltzmann Method*. Studienarbeit, 2007.

[57] Java. http://java.sun.com/. [Online; zugegriffen 21-Januar-2009].

[58] Junk, M.: *A Finite Difference Interpretation of the Lattice Boltzmann Method*. Num. Meth. Part. Diff. Equations, 17:383–402, 2001.

[59] Junk, M. und Klar, A.: *Discretizations for the Incompressible Navier–Stokes Equations Based on the Lattice Boltzmann Method*. SIAM J. Sci. Comput., 22(1):1–19, 2000, ISSN 1064-8275.

[60] Junk, M., Klar, A. und Luo, L.: *Asymptotic analysis of the lattice Boltzmann equation*. Journal Comp. Phys., 210:676, 2005.

[61] Kandhai, D., Vidal, J. E., Hoekstra, A., Hoefsloot, P., Iedema, P. und Sloot, P.: *Lattice-Boltzmann and Finite Element Simulation of Fluid FLow in a SMRX Static Mixer Reactor*. Int. J. Num. Meth. Fluids, 31:1019–1033, 1999.

[62] Karypis, G. und Kumar, V.: *METIS - A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*. http://glaros.dtc.umn.edu/gkhome/views/metis, 1998.

[63] Kaushik, N.: *Development of an octree based mesh generator for polyhedral objects for a LB-solver*. Diplomarbeit, 2004.

[64] Kitware: *Visualization ToolKit (VTK)*. http://www.vtk.org. [Online; zugegriffen 10-Juli-2008].

[65] Kollmannsberger, S., Geller, S., Düster, A., Sorger, C., Rank, E., Tölke, J. und Krafczyk, M.: *Fixed-grid Fluid-Structure interaction in two dimensions based on a partitioned Lattice Boltzmann and p-FEM approach*. Int. J. for Numerical Methods in Engineering, 2009.

[66] Krafczyk, M.: *Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung*. Postdoctoral thesis, Lehrstuhl für Bauinformatik, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München, 2001.

[67] Krafczyk, M., Tölke, J., Rank, E. und Schulz, M.: *Two-Dimensional Simulation of Fluid-Structure Interaction using Lattice-Boltzmann Methods*. Computers and Structures, 79:2031–2037, 2001.

[68] Küttler, U. und Wall, W.: *Fixed-point fluid-structure interaction solvers with dynamic relaxation*. Computational Mechanics, 1(43):61–72, 2008.

[69] Küttler, U. und Wall, W.: *Vector Extrapolation for Strong Coupling Fluid-Structure Interaction Solvers*. Journal of Applied Mechanics, 2(76), 2009.

[70] Ladd, A.: *Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1: theoretical foundations*. Journal of Fluid Mechanics, S. 271–285, 1994.

[71] Ladd, A.: *Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2: numerical results*. Journal of Fluid Mechanics, S. 271–311, 1994.

[72] Lai, Y., Lin, C. L. und Huang, J.: *Accuracy and Efficiency Study of Lattice Boltzmann Method for Steady Flow Simulations*. Numerical Heat Transfer Journal, Part B: Fundamentals 39:21–43, 2001.

[73] Lallemand, P. und Luo, L. S.: *Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability*. Physical Review E, 61(6):6546–6562, 2000.

[74] Lallemand, P. und Luo, L. S.: *Lattice Boltzmann method for moving boundaries*. Journal of Computational Physics, 184:406–421, Jan. 2003.

[75] Lee, T. und Lin, C.: *A Characteristic Galerkin Method for Discrete Boltzmann Equation*. J. Comp. Phys., 171:336–356, 2001.

[76] Leydag, S.: *Implementation eines Gitter-Boltzmann Modells zur Simulation bidirektionaler Fluid-Struktur-Wechselwirkungen zwischen Starrkörpern und Newtonschen Fluiden*. Diplomarbeit, 2002.

[77] Löhner, R., Cebral, J., Camelli, F., Baum, J., Mestreau, E. und Soto, O.: *Embedded/Immersed unstructured grid techniques*. Archives Of Computational Methods In Engineering, 14(3):279–301, 2007.

[78] Liere, R., Mulder, J. und Wijk, J.: *Computational Steering*. Future Generation Computer Systems, 12:441–450, 1997.

[79] Lindrud, J.: *RCF - Interprocess communication for C++*. http://www.codeproject.com. [Online; zugegriffen 10-Juli-2008].

[80] Linhao, Z., Shide, F. und Shouting, G.: *Wind-driven ocean circulation in shallow water lattice Boltzmann model* . Advances in Atmospheric Sciences, 22(3):349–358, 2005.

[81] Linxweiler, J.: *Ein Prototyp zur immersiven Betrachtung und interaktiven Manipulation räumlicher Objekte* . Diplomarbeit, 2004.

[82] Lockard, D. P., Luo, L. S., Milder, S. D. und Singer, B. A.: *Evaluation of PowerFLOW for aerodynamic applications*. Journal of Statistical Physics, 107(1/2):423–478, 2002.

[83] Löhner, R., Baum, J. D., Mestreau, E., Sharov, D., Charman, C. und Pelessone, D.: *Adaptive embedded unstructured grid methods*. International Journal for Numerical Methods in Engineering, 60:641–660, 2004.

[84] Lorensen, W. E. und Cline, H. E.: *Marching cubes: A high resolution 3D surface construction algorithm*. In: *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, Bd. 21, S. 163–169, New York, NY, USA, July 1987. ACM Press.

[85] Luo, L. S.: *Consistent Initial Conditions for LBE Simulation*. Computers and Fluids, 35(8-9):855–862, sep-nov 2006.

[86] Mackerle, J.: *Finite element linear and nonlinear, static and dynamic analysis of structural elements: a bibliography*. International Journal for Computer-Aided Engineering, 14(4):347–440, 1997.

[87] Mei, R., Yu, D., Shyy, W. und Luo, L. S.: *Force evaluation in the lattice Boltzmann method involving curved geometry*. Phys. Rev., 65(4), apr 2002.

[88] Mittal, S. und Kumar, V.: *Finite element study of vortex-induced cross-flow and in-line oscillations of a circular cylinder at low Reynolds numbers*. International Journal for Numerical Methods in Fluids, 31:1087–1120, Dez. 1999.

[89] Mittelstaedt, A.: *Entwicklung eines objektorientierten, VTK-basierten 3D-VirtualReality-Strömungssimulators für das Lattice-Boltzmann-Verfahren* . Diplomarbeit, 2004.

[90] Mok, D. P.: *Partitionierte Lösungsansätze in der Strukturdynamik und der Fluid-Struktur-Interaktion*. Dissertation, Institut für Baustatik, Universität Stuttgart, 2001.

[91] Nanelli, F. und Succi, S.: *The lattice Boltzmann equation on irregular lattices*. J. Stat. Phys., 68(3-4), 1992.

[92] Nebel, B. http://www.bernd-nebel.de/bruecken/4_desaster/tacoma/tacoma.html. [Online; zugegriffen 13-Juli-2008].

[93] Newmark, N.: *A numerical method for structural dynamics*. Journal of Engineering Mechanics (ASCE), 85:67–94, 1959.

[94] Nguyen, N. Q. und Ladd, A. J. C.: *Sedimentation of hard-sphere suspensions at low Reynolds number*. Journal of Fluid Mechanics, 525:73–104, Feb. 2005.

[95] Noble, D., Georgiadis, J. und Buckius, R.: *Comparison of Accuracy and Performance for Lattice Boltzmann and Finite Difference Simulations of Steady Viscous Flow*. Int. J. Num. Meth. Fluids, 23:1–18, 1996.

[96] OpenGL. http://de.wikipedia.org/wiki/OpenGL. [Online; zugegriffen 14-April-2009].

[97] O'Rourke, J.: *Computational Geometry in C*. Cambridge University Press, 2001.

[98] Pahl, P. J.: *Bauinformatikskript - Theoretische Methoden IV*, 1991. TU Berlin.

[99] Python. http://www.python.org/. [Online; zugegriffen 14-April-2009].

[100] Qian, Y. H., d'Humières, D. und Lallemand, P.: *Lattice BGK models for Navier-Stokes equation*. Europhysics Letters, 17:479–484, Feb. 1992.

[101] Rank, E., Bröker, H., Düster, A., Krause, R. und Rücker, M.: *The p-version of the finite element method for structural problems*. In: Stein, E. (Hrsg.): *Error-controlled Adaptive Finite Elements in Solid Mechanics*, Kap. 8, S. 263–307. John Wiley & Sons, 2002.

[102] Rheinländer, M.: *A Consistent Grid Coupling Method for Lattice-Boltzmann Schemes*. J. of Statistical Physics, 121, 2005.

[103] Schäfer, M. und Turek, S.: *Benchmark computations of laminar flow around cylinder*. In: Hirschel, E. H. (Hrsg.): *Notes on Numerical Fluid Mechanics*, Bd. 52, S. 547–566. Vieweg-Verlag, 1996.

[104] Schauer, M.: *Fluid Struktur Interaktion auf Basis der Lattice Boltzmann und Finite Elemente Methode in 3D*. Studienarbeit, 2007.

[105] Schikuta, E. und Message-Passing-Interface-Forum: *MPI: A Message-Passing Interface Standard*. Techn. Ber., University of Tennessee, Knoxville, Tennesee, 1994.

[106] Schiller, L. und Naumann, A.: *Über die grundlegenden Berechnungen bei der Schwerkraftaufbereitung*. Zeitschrift Des Vereines Deutsch. Ing., 77(12):318–320, 1933.

[107] Schneider, K. J.: *Bautabellen für Ingenieure*. Werner, Neuwied, 2006, ISBN 3804152287.

[108] Scholz, D.: *An anisotropic p-adaptive method for linear elastostatic and elastodynamic analysis of thin-walled and massive structures*. Dissertation, Lehrstuhl für Bauinformatik, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München, 2006.

[109] Scholz, D., Düster, A. und Rank, E.: *Model-adaptive structural FEM computations for fluid-structure interaction*. In: *Proceedings of the Third M.I.T. Conference on Computational Fluid and Solid Mechanics*, Cambridge, USA, 2005.

[110] Schroeder, W., Martin, K. und Lorensen, B.: *The Visualization Toolkit, Third Edition*. Kitware Inc., 2002, ISBN 1930934122.

[111] Schulz, M., Krafczyk, M., Tölke, J. und Rank, E.: *Parallelization Strategies and Efficiency of CFD Computations in complex geometries using Lattice-Boltzmann methods on High-Performance Computers*. In: Breuer, M., Durst, F. und Zenger, C. (Hrsg.): *Proceedings of the 3rd International FORTWIHR Conference on HPSEC*, S. 115–122. High-Performance Scientific and Engineering Computing, 12.-14. März 2002.

[112] Shepard, D.: *A two dimensional interpolation function for regularly spaced data*. In: *Proc. 23d National Conf. of the Association for Computing Machinery*, S. 517–524, Princeton, NJ, USA, 1968. ACM.

[113] Shi, X. und Lim, S.: *A LBM-DLM/FD method for 3D fluid-structure interactions*. Journal of Computational Physics, 226(2):2028–2043, 2007.

[114] Shi, X., Lin, J. und Yu, Z.: *Discontinuous Galerkin spectral element lattice Boltzmann method on triangular element*. Int. J. Num. Meth. Fluids, 42:1249–1261, 2003.

[115] Shi, X. und Phan-Thien, N.: *Distributed Lagrange multiplier/fictitious domain method in the framework of lattice Boltzmann method for fluid-structure interactions*. Journal of Computational Physics, 206(1):81–94, 2005.

[116] Spinczyk, O.: *Vortrag am CAB: Was bringt die Aspektorientierte Programmierung?*, 2006.

[117] Spinczyk, O., Lohmann, D. und Urban, M.: *AspectC++: an AOP Extension for C++*. Software Developer's Journal, S. 68–76, 2005.

[118] STL. `http://en.wikipedia.org/wiki/STL_(file_format)`. [Online; zugegriffen 23-Juli-2009].

[119] Succi, S.: *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, Oxford, 2001.

[120] Sui, Y., Chew, Y. T., Roy, P. und Low, H. T.: *A hybrid immersed-boundary and multi-block lattice Boltzmann method for simulating fluid and moving-boundaries*. International Journal for Numerical Methods in Fluids, 53(11):1727–1754, 2007.

[121] Sun: *Remote Method Invocation (RMI)*. `http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp`. [Online; zugegriffen 10-Juli-2008].

[122] Szabó, B. und Babuška, I.: *Finite element analysis*. John Wiley & Sons, 1991, ISBN 0-471-50273-1.

[123] Szabó, B., Düster, A. und Rank, E.: *The p-version of the Finite Element Method*. In: Stein, E., Borst, R. de und Hughes, T. J. R. (Hrsg.): *Encyclopedia of Computational Mechanics*, Bd. 1, Kap. 5, S. 119–139. John Wiley & Sons, 2004.

[124] Tallec, P. le und Mouro, J.: *Fluid Structure Interaction with Large Structural Displacements*. Computer Methods in Applied Mechanics and Engineering, 190(24-25):3039–3068, 2001.

[125] Tcl. http://de.wikipedia.org/wiki/Tcl. [Online; zugegriffen 14-April-2009].

[126] Tezduyar, T., Sathe, S., Keedy, R. und Stein, K.: *Space-time finite element techniques for computation of fluid-structure interactions*. Computer Methods in Applied Mechanics and Engineering, 195:2002–2027, 2006.

[127] The Object Management Group: *Common Object Request Broker Architecture (CORBA) Specification, Version 3.1*. http://www.omg.org/spec/CORBA/3.1. [Online; zugegriffen 10-Juli-2008].

[128] Thürey, N.: *A single-phase free-surface Lattice-Boltzmann Method*. Diplomarbeit, 2003.

[129] Tölke, J., Freudiger, S. und Krafczyk, M.: *An adaptive scheme using hierarchical grids for lattice Boltzmann multi-phase flow simulations*. Computers and Fluids, 35:820–830, Jan. 2006.

[130] Tölke, J., Krafczyk, M. und Rank, E.: *A Multigrid-Solver for the Discrete Boltzmann Equation*. Journal of Statistical Physics, 107 (1/2):573–591, 2002.

[131] Tölke, J., Krafczyk, M., Schulz, M. und Rank, E.: *Discretization of the Boltzmann equation in velocity space using a Galerkin approach*. Comp. Phys. Comm., 129:91–99, 2000.

[132] Tölke, J., Krafczyk, M., Schulz, M., Rank, E. und Berrios, R.: *Implicit discretization and nonuniform mesh refinement approaches for FD discretizations of LBGK Models*. Int. J. of Mod. Phys., 9 (8):1143–1157, 1998.

[133] Turek, S.: *On Discrete Projection Methods for the Incompressible Navier-Stokes Equations: An Algorithmical Approach*. Comput. Methods Appl. Mech. Engrg., 143:271–288, 1997.

[134] Turek, S.: *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*. Springer, 1999.

[135] Turek, S. und Rannacher, R.: *A simple nonconforming quadrilateral Stokes element*. Numer. Meth. Part. Diff. Equ., 8:97–111, 1992.

[136] Walhorn, E., Kölke, A., Hübner, B. und Dinkler, D.: *Fluid-structure coupling within a monolithic model involving free surface flows*. Computers & Structures, 83(25-26):2100–2111, 2005.

[137] Wall, W., Genkinger, S. und Ramm, E.: *A strong coupling partitioned approach for fluid-structure interaction with free surfaces*. Computers & Fluids, 36(1):169–183, 2007.

[138] Wall, W., Gerstenberger, A., P., G., Förster, F. und E., R.: *Large Deformation Fluid-Structure Interaction — Advances in ALE Methods and New Fixed Grid Aporaches*. In: Bungartz, H. und Schäfer, M. (Hrsg.): *Fluid-Structure Interaction, Modelling, Simulation and Optimisation*, Bd. 53 d. Reihe *Lecture Notes in Computational Science and Engineering*, S. 294–335. Springer, 2006, ISBN 978-3540345954.

[139] Wolf-Gladrow, D.: *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*. Springer-Verlag, Berlin, Heidelberg, 2000.

[140] Yu, D.: *Viscous Flow Computations with the Lattice Boltzmann equation method*. Dissertation, Univ. of Florida, 2002.

[141] Yu, D., Mei, R. und Shyy, W.: *A multi-block lattice Boltzmann method for viscous fluid flows*. Int. J. Numer. Methods Fluids, 39(2):99–120, 2002.

[142] Zhou, J.: *A centered scheme for force terms in lattice Boltzmann methods*, 2002.

[143] Zhou, J.: *Lattice Boltzmann Methods for Shallow Water Flows*. Springer, Berlin, 2003, ISBN 3540407464.

[144] Zienkiewicz, O. C.: *The finite element method in engineering science*. London, New York McGraw-Hill, 1971, ISBN 0070941386.